Adabas®

Installation Manual (OS/390, z/OS, OS IV/F4)

Manual Order Number: ADA741-010ZOF

This document applies to Adabas Version 7.4 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover or to the following e-mail address:

Documentation@softwareag.com

TABLE OF CONTENTS

ABOUT THIS MANUAL

This manual provides information for installing and configuring Adabas version 7.4 on the following systems:

- IBM OS/390, z/OS
- Fujitsu OS IV/F4 MSP

See page 5 for the specific operating system levels supported by this Adabas release.

Operating system requirements are provided, as well as procedures for installing Adabas, for connecting Adabas to TP monitor subsystems such as Software AG's Com-plete or IBM's CICS, and for adding new I/O devices.

Note:
Dataset names starting with DD are referred to in Adabas manuals with a slash separating the DD from the remainder of the dataset name to accommodate VSE/ESA dataset names that do not contain the DD prefix. The slash is not part of the dataset name.

Who Should Read and Use This Manual

This manual is intended for those who plan or perform Adabas installation, and for those who manage or maintain an Adabas database system (such as database administrators and systems programming personnel).

How the Manual is Organized

This manual is organized as follows:

Units	Content
Chapter 1	provides pre-installation information for all operating systems: supported environments, a way to check your Adabas version using a program.
Chapters 2–3	provides the job control statements or EXEC commands, step-by-step procedures, and system-related features and considerations for installing Adabas in a separate chapter for each operation system: 2 – OS/390 and z/OS 3 – OS IV/F4 (FACOM)
Chapter 4	tells you how to install Adabas with Com-plete, AIM/DC, CICS (including command-level link and high-performance stub information), IMS, and other teleprocessing monitors supported by Adabas.
Chapter 5	describes device types, VSAM usage for Adabas, special device type considerations and information about using ZAPs to include new devices in the system configuration.
Chapter 6	tells you how to install the demo version of Adabas Online System (AOS) included with Adabas.
Chapter 7	tells you how to install the Adabas Recovery Aid.
Chapter 8	explains the procedure for installing the error handling and message buffering feature and optional PIN routines.
Chapter 9	tells you how to install the TCP/IP database link.
Chapter 10	describes the dump formatter ADAFDP.
Appendix A	reproduces the delivered EBCDIC/ASCII translation tables.
Appendix B	tells you how to calculate the number of relative Adabas block numbers (RABNs); that is, the number of Adabas Associator, Data Storage, Work, Temp, Sort, and CLOG/PLOG blocks available, based on the number of device cylinders allocated.
Appendix C	is a glossary of terms

Other Manuals You May Need

The following Software AG manuals are referred to in this manual; they may be needed when installing Adabas:

- *Adabas Release Notes* (ADAvrs-008IBB)
- *Adabas Operations Manual* (ADAvrs-110IBB)
- *Adabas DBA Reference Manual* (ADAvrs-030IBB)
- *Adabas Triggers and Stored Procedures Manual* (ADAvrs–032IBM)
- *Adabas Command Reference Manual* (ADAvrs–050IBB)
- *Adabas Messages and Codes* (ADAvrs-060IBB)
- *Adabas Utilities Manual* (ADAvrs-080/81IBB—two volumes)
- *Adabas Security Manual* (available only on written request from an authorized user site representative)

For Software AG's System Maintenance Aid (SMA) information, see the manual *System Maintenance Aid Manual* (SMAvrs-030IBB).

The "vrs" portion of the order numbers varies with product releases and manual editions. For Software AG manuals, "vrs" is the version, revision, and system maintenance (SM) level of the product for which the manual was revised or updated.

Depending on your system configuration, the following manuals are also referred to in this manual and may be required for installing or maintaining Adabas:

For **IBM OS/390** and **z/OS** operating systems:

- *Extended Addressability Guide* (GC28-1652)

If installing Adabas on an OS/390 or z/OS system using VSAM files, the following **IBM DFSMS/MVS** manuals should also be available:

- *Access Method Services for VSAM*
- *Using Data Sets*

For **IBM CICS/ESA** environments:

- *System Definition Guide* (for CICS version 3.2 on OS/390 systems)

1

SUPPORTED ENVIRONMENTS

Before attempting to install Adabas, ensure that the host operating system is at the minimum required level.

Adabas version 7.4 is available for the following operating system environments:

- OS IV/MSP 20, EX, and AE (Address Extension 31-bit mode), and AF V10L10.
- OS/390 version 2 releases 6, 7 , 8, 9, and 10. Release 10 is required for 64-bit support.
- z/OS version 1 releases 1, 2, and 3

Software AG provides Adabas support for the operating system versions supported by their respective manufacturers. Generally, when an operating system provider stops supporting a version of an operating system, Software AG will stop supporting that operating system version.

Although it may be technically possible to run a new version of Adabas on an old operating system, Software AG cannot continue to support operating system versions that are no longer supported by the system's provider.

If you have questions about support, or if you plan to install Adabas on a release, version, or type of operating system other than those included in the list above, consult your Adabas technical support facility to determine whether support is possible, and under what circumstances.

2

INSTALLING UNDER OS/390 AND z/OS

This chapter tells you how to prepare for and install Adabas on IBM OS/390 and z/OS operating systems.

For information about the prerequisite system levels supported by Adabas, refer to page 5.

Installation Checklist

The following list provides an overview and procedure for installing Adabas on an OS/390 or z/OS system. The rest of this chapter provides details for the steps in the list.

Basic Installation Steps

Step 1. Allocate DASD space for the Adabas libraries.

The libraries are restored from the installation tape. Refer to the section **Disk Space Requirements for Libraries** on page 9.

Step 2. Allocate DASD space for the Adabas database.

For better performance, distribute the database files over multiple devices and channels. Refer to the section **Disk Space Requirements for the Database** on page 10.

Step 3. Specify the address space for running the Adabas nucleus.

Refer to the section **Adabas Nucleus Address Space Requirements** on page 11.

Step 4. Restore the Adabas libraries from the installation tape.

Use the tape positioning information that accompanies the tape. Refer to the section **Installing the Release Tape** on page 12.

Step 5. Install the Adabas SVC temporarily or permanently.

Refer to the section **Initializing the Adabas Communication Environment** on page 13.

Database Installation Steps

Steps 6–15 require changes to the setup definitions (“customizing”) as described under the section **Installing an Adabas Database** on page 23.

6 Allocate and format the Adabas database with the ADAFRM utility job.

7 Define the global database characteristics with the ADADEF utility job.

8 Load the demonstration (demo) files with the ADALODE, ADALODV, and ADALODM jobs.

9 Start the Adabas nucleus and test the Adabas communications with the ADANUC job.

10 If appropriate, test Adabas address space communications by running ADAREP in MULTI mode with the CPEXLIST parameter.

11 If appropriate, load the Adabas Online System (AOS) selectable unit into a Natural system file by running the AOSINPL job. Alternatively, install the AOS demo version delivered with Adabas: see page 139.

12 Terminate the Adabas nucleus with an ADAEND operator command using the OS Modify command (“F”).

13 Back up the database by running the ADASAV utility job.

14 Insert the ADARUN defaults by running the DEFAULTS job.

TP Monitor Installation

15 Install the required TP link routines for Adabas (see the chapter **Installing Adabas with TP Monitors** starting on page 57).

Preparing to Install Adabas

The major steps in preparing for Adabas installation are

- checking for the correct prerequisite system configuration; and
- allocating disk and storage space.

Disk Space Requirements for Libraries

The minimum 3390 disk space requirements for the Adabas libraries as follows:

Libraries	3390 Cylinders	3390 Tracks	Directory Blocks
Load	8	120	40
Source	3	45	20
JCL	1	15	20

Note:
You can isolate user programs from the Adabas load library by creating a separate load library that contains only those modules needed to execute user programs in MULTI-user mode and linked with ADAUSER. For Version 7.4, the modules required by user programs are:

ADAIOR *ADAMLF*
ADAIOS *ADAPRF*
ADALNK *ADARUN*

Datasets Required for SMH Support

The Software AG WAL library is required if you intend to use Software AG's System Management Hub (SMH).

- The following dataset must be loaded and included in the steplib concatenation:

```
WAL741.MVSLOAD
```

- The remaining datasets are required when installing the System Management Hub:

```
WAL741.MVSLOAD
WAL741.MVSSRCE
WAL741.MVSJOBS
WAL741.ALLAZIP
WAL741.MVSSARG
```

Datasets Required for UES Support

The Software AG internal product libraries (BTE – basic technologies; and APS – porting platform) are required if you intend to enable a database for universal encoding service (UES) support. These libraries are now delivered separately from the product libraries.

For UES support, the following libraries must be loaded and included in the steplib concatenation:

```
BTE312.MVSLDnn
APS271.MVSLDnn
```

—where “nn” is the load library level. If the library with a higher level number is not a full replacement for the lower level load library(s), the library with the higher level must precede those with lower numbers in the steplib concatenation.

Note:
If you are using an Adabas load library prior to version 7.2.2, it contains internal product libraries with an earlier version number and must be ordered below the current internal product libraries in the steplib concatenation.

Also for UES support, the following library must be loaded and included in the session execution JCL:

```
BTE312.MVSECSO
```

For information about setting up connections to UES-enabled databases through Entire Net-Work and ADATCP, see the chapter **Connecting UES-Enabled Databases** starting on page 107.

Disk Space Requirements for Internal Product Datasets

The minimum disk space requirements on a 3390 disk for the internal product libraries delivered with Adabas version 7.4 are as follows:

Libraries	3390 Cylinders	3390 Tracks	Directory Blocks
BTE312.MVSLD00	2	30	5
BTE312.MVSECSO	12	180	150
APS271.MVSLD00	5	75	55

Disk Space Requirements for the Database

The actual database space needed by Adabas depends on user requirements. The minimum 3390 disk space requirements for the database are as follows:

Database Component	3390 Cylinders Required	3390 Tracks Required
ASSOR1 (Associator)	20	300
DATAR1 (Data Storage)	60	900
WORKR1 (Work space)	15	225
TEMPR1 (temporary work space)	15	225
SORTR1 (sort work space)	15	225

Adabas Nucleus Address Space Requirements

The typical Adabas nucleus requires at least 800–1024 kilobytes to operate. The size of the nucleus address space may need to be larger, depending on the ADARUN parameter settings. Parameter settings are determined by the user.

Installing the Release Tape

Copying the Tape Contents to Disk

If you are using SMA, please refer to the chapter **Installing Software AG Products with SMA** in the *System Maintenance Aid* manual.

If you are not using SMA, please follow the instructions below.

This section explains how to:

- Copy dataset COPY.JOB from tape to disk.
- Modify the dataset according to your local naming conventions.

You can use the modified dataset to copy all datasets from tape to disk. You will then need to perform the individual install procedure for each component.

Step 1a: Copy dataset COPY.JOB from Tape to Disk (SMA-jobnumber Tnnn)

The dataset COPY.JOB (label 2) contains the JCL to unload all other existing datasets from tape to disk. To unload COPY.JOB, use the following sample JCL:

```
//SAGTAPE  JOB SAG,CLASS=1,MSGCLASS=X
//* - - - - - - - - - - - - - - - - - -
//COPY    EXEC PGM=IEBGENER
//SYSUT1  DD  DSN=COPY.JOB,
//            DISP=(OLD,PASS),
//            UNIT=(CASS,,DEFER),
//            VOL=(,RETAIN,SER=<Tnnnnn>),
//            LABEL=(2,SL)
//SYSUT2  DD  DSN=<hilev>.COPY.JOB,
//            DISP=(NEW,CATLG,DELETE),
//            UNIT=3390,VOL=SER=<vvvvvv>,
//            SPACE=(TRK,(1,1),RLSE),
//            DCB=*.SYSUT1
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//
```

—where

<hilev>	is a valid high level qualifier
<Tnnnnn>	is the tape number
<vvvvvv>	is the desired volser

Step 1b: Modify COPY.JOB to your Local Naming Conventions

There are three parameters to set before you can submit this job:

- Set HILEV to a valid high level qualifier.
- Set LOCATION to a storage location.
- Set EXPDT to a valid expiration date.

Step 1c: Submit COPY.JOB

Submit COPY.JOB to unload all other datasets from tape to your disk.

Initializing the Adabas Communication Environment

This section describes the installation of the Adabas router (ADASVC). The router uses cross-memory services for communication between the Adabas nucleus and the Adabas users.

The Adabas OS/390 or z/OS cross-memory communications service comprises two modules:

- the Adabas router (ADASVC); and
- the Adabas subsystem initialization routine (ADASIR).

ADASIR, executed either during IPL or by the Adabas SVC installation program (ADASIP), initializes the router's operating environment, particularly the ID table.

ADASVC installation can be either temporary or permanent:

- The Adabas SVC can be installed temporarily by executing ADASIP. The SVC is then available only until the next IPL.

  Once installed, the Adabas SVC can be reinstalled temporarily using the ADASIP REPLACE option. Note, however, that no Adabas nucleus can be active during this procedure.

  Note:
  It is necessary to cycle CICS after executing ADASIP to initialize the SVC.

- The Adabas SVC is installed permanently using regular operating systems procedures. The SVC then requires an IPL to become active.

Typically, the Adabas SVC is first temporarily installed using ADASIP. This makes Adabas available immediately without the need to wait for an IPL. Meanwhile, preparations are usually made for permanent installation at the next IPL.

Allocating an SVC Table Entry

Regardless of the installation procedure selected, an available SVC table entry must be allocated to the Adabas router (ADASVC). SVC table entries are defined in the member IEASVCxx of SYS1.PARMLIB.

The SVC table entry in the operating system for an ADASVC must contain the following information:

Offset	Label	Description
0	SVCEP	SVC entry point address.
4	SVCATTR1	Must indicate type 2 SVC (flag bit SVCTP2 set—X'80') or type 3 or 4 SVC (flag bits SVCTP34 set—X'C0'): ADASIR changes a type 1, 5, or 6 SVC to type 2. May indicate that APF-authorization is needed for this SVC (flag bit SVCAPF set—X'08'): if set, all targets and users must be APF-authorized.
6	SVCLOCKS	Must contain all zeros. ADASIR sets SVCLOCKS to zeros.

Subsystem Name Requirements

The subsystem name contained in the four-character field SUBSYS at ADASVC offset X'28' (the default is "ADAB") must be the same as that specified in the IEFSSNxx member of SYS1.PARMLIB. If the name is not the same, ADASIR ends with an ADAS12 message and condition code of 2, and Adabas is not usable.

Page-Fixing the Adabas SVC

If the Adabas SVC is to reside in the fixed LPA, add an entry to an IEAFIXxx member of SYS1.PARMLIB.

Initializing the Adabas SVC

The Adabas SVC should be initialized with ADASIP/ADASIR in order to guarantee full functioning of all Adabas nuclei in the "cluster".

Router Installation Overview

Temporary Router Installation (SMA-jobnumber I011)

Once you have restored the Adabas installation tape, use a local editor facility to customize the job JCLLINK (used to link ADASIR, ADASIP, and ADASVC) as follows:

Step 1. Link ADASIP into an APF-authorized library as an authorized module.

Step 2. Link ADASIR and ADASVC into APF-authorized libraries.

- Place ADASVC in an APF-authorized library in order to run ADASIP.
- Place ADASIR in an APF-authorized library concatenated to SYS1.LINKLIB defined in source member LNKLSTxx located in SYS1.PARMLIB.

Step 3. Execute ADASIP to install the SVC.

Customize and run the job ADASIP to dynamically add the Adabas SVC without an IPL.

Permanent Router Installation (SMA-jobnumber I010)

Permanent router installation comprises the following steps:

Step 1. Link the Adabas SVC (ADASVC) which has been renamed according to the SVC routine renaming rules (for example, type 3 SVCs must have names of IGC00nnn, where “nnn” is a “signed” decimal SVC number) into SYS1.LPALIB as a permanent step for ADASIR.

Step 2. Link ADASIR into SYS1.LINKLIB or into an APF-authorized library concatenated to SYS1.LINKLIB with the LNKLSTxx member of SYS1.PARMLIB (note that ADASIR is neither reentrant nor refreshable, and therefore should not be linked into SYS1.LPALIB).

Step 3. Customize and run the job JCLUPDT to add a new entry with the format

```
SUBSYS SUBNAME(cccc) INITRTN(ADASIR) INITPARM('parameters') comments
```

—into the active member IEFSSNxx in SYS1.PARMLIB for ADASIR. This allows the re-IPL to install the Adabas SVC automatically. See the section **Executing ADASIR** on page 20 for more information and an explanation of the entry format.

Step 4. IPL OS/390 or z/OS with the CLPA option to install and initialize the Adabas communication environment.

Using ADASIP for Temporary Installations

ADASIP Functions

ADASIP performs the following functions:

- Acquires memory in the specified CSA subpool for the Adabas SVC and a subsystem communication vector table (SSCT);
- Loads the Adabas SVC into the acquired CSA space;
- Modifies the SVC table entry as required by the Adabas SVC;
- Optionally deletes an SSCT for the same subsystem name from the SSCT chain;
- Adds the new SSCT to the SSCT chain;
- Invokes the ADASIR program;
- If any error is detected, ADASIP backs out all completed activities and terminates operation with a user ABEND specifying the error.

The following JCL links ADASIP, located in ADABAS.Vvrs.LOAD, into an APF-authorized library as an authorized module:

```
//LNKSIP  EXEC PGM=IEWL
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD SPACE=(CYL,(1,1)),UNIT=SYSDA
//ADALIB    DD DSN=ADABAS.Vvrs.LOAD,DISP=SHR
//SYSLMOD   DD DSN=apflibname,DISP=SHR
//SYSLIN    DD *
 INCLUDE ADALIB(ADASIP)
 SETCODE AC(1)
 NAME ADASIP(R)
```

ADASIP Parameters

The parameters are of the form

CONSID=c**,IDTSPL=**i**,LEAVE=**l**,NRIDTES=**n**,REPLACE=**r**,SUBSYS=**su**,**
SVCNR=svcn**,SVCSPL=**svcs

—where

c is the console ID to which operator messages are written: 1-255 (default: 1).

i is the ID table subpool: see the ADASIR IDTSPL parameter for details.

l	indicates whether ADASIR should leave message ADAS11 or ADAS12 on the operator console: see the ADASIR LEAVE parameter for details.
n	is the number of ID table entries: see the ADASIR NRIDTES parameter for details.
r	indicates whether an existing SSCT for the same subsystem name is replaced: "Y" for yes or "N" for no ("N" is the default). Use this option to replace any type of Adabas SVC (for example, when installing a new SVC version).
su	is the subsystem name. This parameter is required, and should always be specified as "ADAB" unless specified otherwise by Software AG.
svcn	is the Adabas SVC number: see the ADASIR SVCNR parameter for details.
svcs	is the Adabas SVC and SSCT subpool: 228 for fixed CSA or 241 for pageable CSA (default: 241).

The following are valid ADASIP parameter abbreviations:

Parameters	Abbreviations
CONSID=	C=
IDTSPL=	I=
LEAVE=	L=
NRIDTES=	N=
REPLACE=	R=
SUBSYS=	SU=
SVCNR=	SVCN=
SVCSPL=	SVCS=

All parameters are optional except SUBSYS and SVCNR. If specified, the parameters IDTSPL, LEAVE, NRIDTES, SUBSYS, and SVCNR are passed to ADASIR without being verified.

Executing ADASIP

JCL similar to the following should be used to execute ADASIP:

```
// EXEC PGM=ADASIP,PARM=parameters
//STEPLIB DD ...
//SVCLIB DD ...
//SIRLIB DD ...
```

The dataset defined by the STEPLIB DD statement must be an APF-authorized library containing the APF-authorized program ADASIP. Since ADASIP is neither reentrant nor refreshable, the dataset cannot be SYS1.LPALIB.

The dataset defined by the SVCLIB DD statement must be an APF-authorized library containing the Adabas SVC with either the name or alias “ADASVC”.

The dataset defined by the SIRLIB DD statement must contain the ADASIR program. Since ADASIR is neither reentrant nor refreshable, the dataset may **not** be SYS1.LPALIB.

ADASIP terminates with a U0481 ABEND if the parameter input is incorrectly specified.

The IBM job control convention for continuing the PARM parameter is:

```
// EXEC PGM=ADASIP,PARM=('parameters ....',             X
// 'parameters')
```

—where “X” in column 72 is a continuation character. The following restrictions also apply to JCL statements:

- A comma is required after the end-quote on a line that is to be continued.
- A non-blank continuation character is required in column 72 of each line that is to be continued, and the continuation line must start within columns 4–16.
- A comma is **not** permitted between the last parameter and the end-quote on the line to be continued because JCL automatically inserts a comma between parameters when concatenating continuation strings:

```
// ...PARM=('CONSID=3',                                X
//       'SUBSYS=ADAB',                                X
//       'SVCNR=249')
```

—results in an equivalent line of

```
CONSID=3,SUBSYS=ADAB,SVCNR=249
```

Using ADASIR

ADASIR Functions

The ADASIR program is invoked

- by the ADASIP program to temporarily install; or
- by OS/390 or z/OS to permanently install the Adabas SVC.

ADASIR receives control during either master scheduler initialization or ADASIP execution. The operator is prompted for any value that has been incorrectly zapped or assembled (refer to the *Adabas Messages and Codes* manual for specific message descriptions). If an error is found during the processing of parameters specified in the IEFSSNxx member or passed by ADASIP, the operator is prompted for all of the values.

If the SVC table entry is incorrect, ADASIR prompts the operator for permission to change the entry (if SVCTAB=P, the default, is specified). If any errors are detected, they must be corrected and either another IPL must be done or ADASIP must be rerun before the Adabas SVC can be used.

Relinking ADASIR

The ADASIR module must be linked into an APF-authorized library.

The following JCL links ADASIR, located in ADABAS.Vvrs.LOAD, into SYS1.LINKLIB:

```
//LNKSIR  EXEC PGM=IEWL
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD SPACE=(CYL,(1,1)),UNIT=3350
//ADALIB    DD DSN=ADABAS.Vvrs.LOAD,DISP=SHR
//SYSLMOD   DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN    DD *
 INCLUDE ADALIB(ADASIR)
 NAME ADASIR(R)
```

ADASIR Parameters

The following are the descriptions of and defaults for the ADASIR parameters:

IDTSPL=i**,LEAVE=**l**,NRIDTES=**n**,SVCNR=**svcn**,SVCTAB=**svct

—where the variable value

i	is the ID table subpool: 228 for fixed CSA or 241 (the default) for pageable CSA.
l	indicates whether message ADAS11 or ADAS12 are to be left on the operator console: “Y” for yes or “N” (the default) for no.
n	is the ID table entry count, which can range from “1” to a maximum specified at offset X’146’ in the CSECT IEAVESVT of the OS/390 or z/OS nucleus (see section **Requirements for Cross-Memory Services** on page 26).
svcn	is the Adabas SVC number (200–255).
svct	indicates whether or not the operator should be prompted for permission to update the SVC table entry. Enter “P” (the default) to receive a prompt, or “N” for no prompt. “P” is recommended if a possibility exists that the SVC table entry will not be what ADASIR expects.

The following are valid abbreviations for ADASIR parameters:

Parameters	Abbreviations
IDTSPL=	I=
LEAVE=	L=
NRIDTES=	N=
SVCNR=	SVCN=
SVCTAB=	SVCT=

Executing ADASIR

Note:
The ADASIR module must be linked into an APF-authorized library. See page 19.

To prepare for permanent SVC installation, an entry must be made in either a new or existing member having the name IEFSSNxx in SYS1.PARMLIB. This entry is an 80-character record with the following format:

```
SUBSYS SUBNAME(cccc) INITRTN(ADASIR) INITPARM('parameters') comments
```

—where

cccc	is the 1- to 4-character subsystem name. This name and the name specified in the Adabas SVC at offset X'28' must be the same. The name provided in the SVC is "ADAB"; any other name must first be zapped into the SVC before being specified for 'cccc'.
'parameters'	if there is more than one parameter, values must be enclosed in single quotation marks and a comma placed between the parameters.
comments	are optional and must be preceded by at least one space.

If the subsystem name does not match, ADASIR ABENDs with an ADAS12 message and condition code of 2; the Adabas OS/390 or z/OS communication environment is not initialized. Re-IPL OS/390 or z/OS, specifying SSN=xx if necessary. If this is the first IPL with a type 3 or 4 Adabas SVC, specify CLPA as one of the SET parameters.

If an error is encountered while processing any of the parameters obtained from the IEFSSNxx member or passed from ADASIP (message ADAS05), the operator is prompted to reenter all of the parameters. If the SVC table entry is not correct (message ADAS09) then, depending on the value of the SVCTAB parameter, either the operator is prompted (message ADAS10) for permission to change the SVCTAB parameter, or it is simply changed (message ADAS15).

A version 6.2 ADASIP/ADASIR can be used to install a version 7 Adabas SVC (and the reverse). However, Software AG recommends that you use the same release of ADASIP/ADASIR to install the SVC/router as the SVC/router itself.

The ADASIR messages and their meanings are described in the *Adabas Messages and Codes* manual.

Relinking the SVC for Temporary Installation

Link the Adabas SVC with the name or alias “ADASVC” into an APF-authorized library. ADASVC must be linked with AMODE=31 and RMODE=24, the default.

The following example shows how to link the SVC:

```
//      (job card)
//LKED      EXEC  PGM=IEWL,
//          PARM='XREF,LIST,LET,NCAL,RENT,REUS'
//SYSPRINT  DD SYSOUT=X
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD DSN=SYS1.LINKLIB,DISP=SHR       <--Target loadlib
//ADALIB    DD DSN=user.loadlib,DISP=SHR       <--ADASVC loadlib
//SYSLIN    DD *
 MODE AMODE(31),RMODE(24)
 INCLUDE ADALIB(ADASVC)
 NAME ADASVC(R)
/*
```

Note:
If the SVC is linked with a name other than “ADASVC” when preparing to upgrade to a permanent installation, the SVC must have an alias of “ADASVC”. When dynamically loading the Adabas SVC, ADASIP searches for the module “ADASVC” in the library specified by the SVCLIB DD statement.

Relinking the SVC for Permanent Installation

Software AG recommends using a type 3 or 4 SVC for the Adabas SVC.

SVC types 1 and 6 are not supported.

For a Type 2 SVC

1. If the Adabas SVC is to be type 2, link it into SYS1.NUCLEUS as the system nucleus IEANUC0x.

   This nucleus must contain an SVC table entry for “an enabled type 2 SVC”, which must be defined during SYSGEN.

2. Then include linkage editor control statements similar to the following with those needed to link a nucleus:

```
CHANGE ADASVC(IGCnnn) ---> nnn is the SVC number in decimal
INCLUDE ADALIB(ADASVC) ---> ADALIB contains the Adabas SVC
```

For a Type 3 or 4 SVC

To install the Adabas SVC as type 3 or 4, link the Adabas SVC with the appropriate name into SYS1.LPALIB. ADASVC must be linked with AMODE=31 and RMODE=24 (the default).

The following example shows how to relink the SVC:

```
//      (job card)
//LKED      EXEC  PGM=IEWL,
//          PARM='XREF,LIST,LET,NCAL,RENT,REUS'
//SYSPRINT  DD SYSOUT=X
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD DSN=SYS1.LPALIB,DISP=SHR          <--Target Loadlib
//ADALIB    DD DSN=ADABAS.Vvrs.LOAD,DISP=SHR     <--ADASVC loadlib
//SYSLIN    DD *
 MODE AMODE(31),RMODE(24)
 CHANGE ADASVC(IGC00nnp)     <- where nn are the first two digits of the SVC in decimal and
 INCLUDE ADALIB(ADASVC)          p is the character corresponding to the x'Cn'
 NAME IGC00nnp(R)       ---->   (n is the last digit of the SVC number in decimal)
*
/*
```

Installing an Adabas Database

Once you have restored the Adabas installation tape and have installed the ADASVC, you can

- migrate an existing Adabas database to the new version; or
- install a new version the Adabas database.

Messages or codes that occur during the installation are described in the *Adabas Messages and Codes* manual; utilities are described in the *Adabas Utilities Manual*.

Migrate an Existing Database (SMA-jobnumber I051)

Use the ADACNV utility to migrate existing databases to new releases of Adabas. See the *Adabas Utilities Manual* for more information.

Install a New Database

Step 1. Allocate and format the Adabas database (SMA-jobnumber I030)

Customize and run the ADAFRM utility job to allocate and format the Adabas database. The following specific items must be customized:

- dataset names for the database and libraries;
- volumes for libraries and datasets for the database;
- space allocation for datasets for the database;
- the Adabas SVC number, the database ID, and database device type(s);
- sizes of the datasets for each ADAFRM statement.

Step 2. Define the global database characteristics (SMA-jobnumber I030)

Customize and run the ADADEF utility job to define the global definition of the database. The following items must be customized:

- dataset names of the database and libraries;
- the Adabas SVC number, the database ID, and database device type(s);
- ADADEF parameters.

Step 3. Load the demonstration (demo) files (SMA-jobnumber I050)

Customize and run the job

- ADALODE to load the sample demo file EMPL;
- ADALODV to load the sample demo file VEHI; and
- ADALODM to load the sample demo file MISC.

For each job, the following items must be customized:

- dataset names for the database and libraries;
- the Adabas SVC number, the database ID, and database device type(s);
- ADALOD parameters.

Step 4. Start the Adabas nucleus and test the Adabas communications (SMA-jobnumber I040)

Customize and run the job ADANUC to start up the Adabas nucleus. The following items must be customized:

- dataset names for the database and libraries;
- the Adabas SVC number, the database ID, and database device type(s);
- ADANUC parameters.

Step 5. Test Adabas address space communications, if appropriate

Customize and run the job ADAREP in MULTI mode with the CPEXLIST parameter to test Adabas address space communications. The following items must be customized:

- dataset names for the database and libraries;
- the Adabas SVC number, the database ID, and database device type(s);
- ADAREP parameters.

Step 6. Load the Adabas Online System selectable unit, if appropriate (SMA-jobnumber I061)

Customize and run the job AOSINPL to load the Adabas Online System (AOS) into a Natural system file using a batch version of Natural 3.1 or above. The following items must be customized:

- dataset names of the database and libraries;
- the Adabas SVC number, the database ID, and device type(s);

- the Natural INPL parameters and system file number.

Alternatively, install the AOS demo version delivered with Adabas: see page 139.

Step 7. Terminate the Adabas nucleus

Communicate with the Adabas nucleus to terminate the session either with an ADAEND operator command using the OS Modify command

F jobname**,ADAEND**

—or

P jobname

—where “jobname” is the started nucleus’ job or task name.

Step 8. Back up the database

Customize and run the ADASAV utility job to back up the database. The following items must be customized:

- dataset names of the database and libraries;
- the Adabas SVC number, the database ID, and device type(s);
- ADASAV parameters.

9 Insert the ADARUN defaults

The member DEFAULTS in the Adabas JCL library can be modified to set the ADARUN defaults. The following items must be customized:

- dataset names for libraries; and
- ADARUN user defaults:
  - device type(s) (default: 3380)
  - SVC number (default: 249)
  - database ID (default: 1)

Customize and run the DEFAULTS job to set the ADARUN defaults using the OS ZAP utility.

Step 10. Install the required TP link routines for Adabas

Refer to the chapter **Installing Adabas with TP Monitors** starting on page 57 for the TP link routine procedure.

SVC Integrity Validation

In the past, the presence of multiple SVCs with the same subsystem ID has resulted in a single ID table being used by different SVCs. This has caused problems, some of them serious (abnormal nucleus termination or corruption of the database).

To eliminate this danger, the version 7 SVC checks the validity by ensuring that the SVC accessing the ID table is the same as the one that was used by ADASIP/ADASIR to initialize the table. If the SVCs are not the same, an ABEND 650 occurs.

ABEND 650 occurs when an incorrect SVC number is specified in the ADARUN parameters for a nucleus. It can occur during Adabas initialization, during the first Adabas call from a user program, or when the ID table is queried by another Software AG server such as Entire Net-Work.

Requirements for Cross-Memory Services

Due to the implementation of cross-memory services in OS/390 and z/OS, the following points should be noted when running an Adabas nucleus in MULTI mode:

1. A maximum of one step of a job can establish the cross-memory environment. This means that a job can include at most one step that is a target (for example, an Adabas nucleus).

2. Cross-memory accesses may not be made to a swapped-out address space. Therefore, the address space of an Adabas nucleus is set to “nonswappable” for the duration of the nucleus session. This can increase the installation’s real storage requirements.

   This behavior is documented in the IBM manual *Extended Addressability Guide*, chapter **Synchronous Cross-Memory Communication**.

3. When a nucleus with an active cross-memory environment terminates either normally or abnormally, the entire address space including any initiator is also terminated.

   The ASID representing this address space is not reassigned until the next IPL. Therefore, you should choose a sufficiently high value for the MAXUSERS parameter in the active IEASYSxx member of SYS1.PARMLIB or—if your system supports it—the RSVNONR parameter in the same member can be adjusted accordingly. Also, the Adabas nucleus should not be stopped and started without good reason.

This is described in the manuals referred to above under the specific topics “Recovery Considerations” and “Resource Management”. Additional information can be found in IBM APARs OZ61154, OZ61741, and OZ67637.

To make its services available to all address spaces in the system, the Adabas nucleus must obtain a system linkage index (LX) from OS/390 or z/OS. The LX is a reserved slot in the linkage space of all address spaces, and permits system-wide linkage to all address spaces.

The number of LXs set aside by OS/390 or z/OS for system use is rather small (usually 165 out of a possible 2048).

Because of the way OS/390 and z/OS use cross-memory services, system LXs obtained by Adabas cannot be returned to OS/390 or z/OS until the next IPL. However, the system that “owns” the LXs can reuse them, even for a different address space. Adabas makes use of this feature by saving used LXs in the ID table, where they are available to future nuclei.

The number of system LXs can be specified in the member IEASYSxx contained in SYS1.PARMLIB, using the NSYSLX parameter. If you change this value, you must perform an IPL to make the change effective.

To determine an appropriate NSYSLX value, consider the following points:

- Some LXs are probably already being used by other system functions. Therefore, the chances of creating an LX shortage for other users is small.
- Adabas requires one system LX for each Adabas nucleus (or any other target) that will be active concurrently. A value of decimal 64 would allow concurrent execution of up to 64 Adabas nuclei or other targets with little chance of restrict other components using LXs.
- Entire Net-Work version 5 uses only one LX and one ID table entry, regardless of how many remote databases it must represent. This is unlike the pseudo-MPM concept of earlier Entire Net-Work versions.
- Whenever ADASIP is executed with the REPLACE option, all LXs saved in the current ID table are lost until the next IPL.

  Likewise, if a session ends either normally with the FORCE operator command or abnormally during ESTAE processing (for example, by an S222 operator cancel or by a S722 “spool limit exceeded” ABEND during a snap dump), the LX also cannot be recovered until the next IPL.

  Any commands sent to these targets receive an S0D6 ABEND. Any attempt to restart the nucleus results in an ADAM98 message with the reason, “DUP ID (LOCAL)”, followed by an ABEND. To resolve both of these problems, restart the nucleus with the ADARUN FORCE=YES and IGNDIB=YES parameters.

The first target that tries to obtain a system LX when none is available ends with an S053 ABEND code and a reason code of 0112. No additional targets can be started until the next IPL.

The only CSA space used by the Adabas version 7 router is the following:

- 96+32*NRIDTES (where NRIDTES is the ADASIR or ADASIP parameter described above) every time either program is successfully executed.
- LEN(ADASVC)+SSCTSIZE (equal to 36 bytes) each time ADASIP executes successfully.

Requirements for Global Resource Serialization

Adabas uses Global Resource Serialization (GRS) to synchronize the execution of Adabas nuclei and utilities at certain points in their processing. It is vital that GRS be set up correctly in the system so that GRS requests by Adabas will be effective.

When setting up GRS, consider the following:

- Adabas uses the GRS macros ENQ and DEQ with systems-wide scope (SCOPE=SYSTEMS) and major name ‘ADABAS’ (QNAME).
- If the database resides on disks that are shared between multiple images of the operating system (multiple LPARs or machines) and Adabas nuclei or utilities may be run against the database from several of these images, make sure that GRS is installed in a way that systems-wide ENQ requests are effective on all of these system images.

Using EXCPVR

CPU usage is reduced considerably by using EXCPVR.

To use EXCPVR, you must

- APF-authorize ADARUN; and
- locate all Adabas modules in APF-authorized libraries.

When placed in an APF-authorized library, ADARUN allows the Adabas nucleus and utilities to use EXCPVR. All Adabas modules except the Adalink must run with RMODE=24 (the default); the Adalink can run with any RMODE.

When the EXEC statement specifies PGM=ADARUN, the AMODE is determined as follows:

- The default AMODE=31 is used unless ADARUN was relinked using either the AMODE=24 EXEC statement parameter or the MODE AMODE(24) linkage editor control statement.
- An AMODE of 31 is changed to 24 before affected macros (data management, for example) are run, and then changed back to 31 thereafter.

Adabas performance can be improved by using EXCPVR to improve channel program translation time. For Adabas to invoke EXCPVR automatically, the Adabas modules must be in an APF-authorized library and ADARUN must be linked with SETCODE AC(1), the default, as shown in the following example:

```
//LINKRUN    EXEC  PGM=IEWL,PARM='REUS'
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//ADALIB    DD DSN=user.loadlib,DISP=SHR
//SYSLMOD   DD DSN=user.loadlib,DISP=SHR      <==APF-AUTHORIZED LIBRARY
//SYSLIN    DD *
  INCLUDE  ADALIB(ADARUN)
  SETCODE  AC(1)
  NAME  ADARUN(R)
```

Creating a Shareable ADARUN

The ADARUN module delivered in the Adabas load library is not reusable. If you need a shareable ADARUN, you will need to relink it with the REUS=YES link-edit attribute.

Linking ADARUN with the reusable option permits several programs running in the same address space to share the same ADARUN and ultimately, the same copy of ADALNK. This is important when it is necessary to have only one Adabas user ID for the different programs, and is also needed if single copies of ADALNK user exits are required.

To create a shareable ADARUN, use the sample job JCLLINRR in the MVSJOBS library to relink it with the REUS attribute.

If both nonreusable and reusable versions of ADARUN are required, they must be located in different load libraries since both must be loadable using the name “ADARUN”.

Storage above 16 MB

Adabas can acquire a number of its required areas, including buffer space, above the 16-MB addressing limit, allowing Adabas to increase the buffer pool size.

To reverse the space allocation to be below the 16-MB limit, set the AMODE value in the MODE statement in the example below to AMODE(24):

```
//LINKRUN    EXEC  PGM=IEWL,PARM='REUS'
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//ADALIB    DD DSN=user.loadlib,DISP=SHR
//SYSLMOD   DD DSN=user.loadlib,DISP=SHR      <=APF-authorized library
//SYSLIN    DD *
  MODE AMODE(24),RMODE(24)
  SETCODE AC(1)
  INCLUDE ADALIB(ADARUN)
  NAME ADARUN(R)
```

In addition, Adabas must be run with a sufficient REGION specification, either on the JOB or EXEC statement or as an installation default. For example:

```
//BIG  JOB  ...,REGION=30M,...
```

Storage above 2 GB (64-Bit)

Real Storage

Adabas can exploit storage occupying real pages above the 2-gigabyte line. This capability allows ADABAS I/Os to use 64-bit real addresses.

Support for 64-bit real storage is available whether you are running APF-authorized (using EXCPVR) or not (using EXCP). The run mode is indicated in the ADAI65 message:

```
ADAI65   EXCPVR IS {BEING | NOT BEING} USED FOR THIS RUN IN ESA64 MODE
```

Support for 64-bit real storage requires either

- OS/390 R10 in ARCHLEVEL=2 (that is, z/architecture mode); or
- z/OS 1.2 or above

—on a processor of the IBM 2064 family with an LPAR greater than 2 gigabytes for real storage allocation.

Virtual Storage

IBM supports 64-bit virtual storage only for z/OS 1.2 or above.

Software AG provides support for IBM's 64-bit virtual storage with a new product Adabas Caching Facility (ACF). Contact your Software AG account representative for more information.

A demo of ADABAS Caching Facility is delivered in the ADA741.ALLINPL file.

Applying ZAPs

Use the OS/390 or z/OS AMASPZAP utility to apply ZAPs in the respective operating system; this method verifies (VER) and replaces (REP) data. The following sample JCL executes AMASPZAP:

```
//ADAZAP    JOB
//STEP1     EXEC  PGM=AMASPZAP
//SYSPRINT  DD SYSOUT=X
//SYSLIB    DD DSN=ADABAS.Vvrs.LOAD,DISP=SHR
//SYSIN     DD *
        (ZAP control statements)
/*
//
```

—where the following are examples of ZAP control statements:

NAME	membername csectname
VER	displacement data
REP	displacement data
IDRDATA	(up to eight bytes of user data)
*	(comment)

Note:
In VER and REP statements, spaces must be used to separate command, displacement, and data. Commas are acceptable data separators; however, commas with spaces or spaces alone are not, and may cause errors.

Adalink Considerations

Note:
For information about connecting a database that is enabled for data conversion using the universal encoding service (UES), see the chapter ***Connecting UES-Enabled Databases*** *starting on page 107.*

User Exit B (Pre-command) and User Exit A (Post-command)

One or two user exits may be linked with an Adalink routine (SMA-jobnumber I088):

- UEXITB receives control **before** a command is passed to a target with the router 04 call.

  Note:
  Special commands emanating from utilities and from Adabas Online System are marked as "physical" calls. These calls must be bypassed in user exits. These calls have X'04' in the first byte (TYPE field) of the command's Adabas control block (ACB). UEXITB must check this byte and return if it is set to X'04'. Be sure to reset R15 to zero on return.

- UEXITA receives control **after** a command has been completely processed by a target, the router, or by the Adalink itself.

  At entry to the exit(s), the registers contain the following:

Reg.	Content
1	Address of the UB If the flag bit UBFINUB is reset, the contents of the halfword at Adabas + X'86' have been moved to UBLUINFO. If those contents are greater than zero, the two bytes starting at UBINFO (UB+X'40') have been set to zero. If UBFINUB is set, no changes can be made to the UB or ACB (except for ACBRSP).
2	Address of a 16-word save area (for ADALNC only)
13	Address of an 18-word save area (for non-CICS Adalink exits)
14	Return address
15	Entry point address: UEXITB or UEXITA

Any registers except register 15 that are modified by the user exits must be saved and restored; the address of a save area for this purpose is in register 13 (or register 2 for ADALNC).

If at return from UEXITB register 15 contains a value other than zero (0), the command is not sent to the target but is returned to the caller. The user exit should have set ACBRSP to a non-zero value to indicate to the calling program that it has suppressed the command: response code 216 is reserved for this purpose.

The UEXITB exit may set the UB field UBLUINFO to any lesser value, including zero; an ABEND occurs if the user exit sets UBLUINFO to a greater value. The UBLUINFO length cannot be changed when any other exit is used; for example, Review or Fastpath.

The user information received by a UEXITA exit may have been modified; this modification may include decreasing its length, possibly to zero, by any of the Adalink user exits.

An Adalink routine can return the following non-zero response codes in ACBRSP:

213 No ID table

216 UEXITB suppressed the command

218 No UB available

At least the following three EQUates, described at the beginning of the source, can be modified before an Adalink routine is assembled. In some Adalink routines, however, the corresponding information can be zapped:

LOGID is the default logical ID, ranging in value from 1 to 65535. The default is 1.

LNUINFO is the length of the user information to be passed to Adalink user exits, ranging in value from 0 to 32767. The default is 0.

SVCNR is the Adabas SVC number; its range of values and the default depend on the operating system. This value can be provided as SYSPARM-value for assembly of the following Adalink routine:
//EXEC PGM=ass,PARM='.....,SYSPARM(svcnr)'

The first 152 (X'98') bytes of all Adabas Adalinks must maintain the following structure:

Offset	Label	Contents	Meaning
00	ADABAS		Entry code
12		CL6'ADALNx'	Program name
18		XL4'yyyymmdd'	Assembly date
1C		A(ZAPTAB)	Address of ZAP table
20	PATCH	XL96'00'	Patch area
80	LNKLOGID	AL2(LOGID)	Default logical ID (default: 1)
82		XL2'00'	Reserved
84	LNKSVC	SVC SVCNR	Executable SVC instruction for Adabas SVC (default: operating-system-dependent)
86	LUINFO	Y(LNUINFO)	Length of user information (default: 0)
88	VUEXITA	V(UEXITA)	Address of user exit after call (weak)
8C	VUEXITB	V(UEXITB)	Address of user exit before call (weak)
90	ADABAS51	CL8'ADABAS51'	IDT ID

ADAUSER Considerations

ADAUSER is a program that links the user to Adabas. It is specific to an operating system and is independent of release level and mode. It can be used in batch and in some TP environments (such as TSO).

ADAUSER operates in the following way:

- ADAUSER contains the entry point ADABAS and should be linked with all user programs that call Adabas. No other programs containing the CSECT or entry point name ADABAS can be linked in these load modules/phases.
- On the first call to Adabas, ADAUSER loads the latest version of ADARUN. This makes the calling process release-independent. Subsequent Adabas calls bypass ADARUN.
- ADARUN processes its control statements. If the ADARUN PROGRAM parameter has the (default) value USER, ADARUN loads ADAIOR and, depending on whether the ADARUN MODE parameter specifies MULTI or SINGLE, loads the appropriate TP Adalink or ADANUC, respectively. This makes the calling process mode-independent.

3

INSTALLING UNDER OS IV/F4

This chapter tells you how to prepare for and install Adabas on Fujitsu OS IV/F4 (including MSP and MSP 20/AE/EX) operating systems.

For information about the prerequisite system levels supported by Adabas, refer to page 5.

Installation Checklist

Note:
The MSP environment now has two tape versions. The newer tape contains MSP/EX support for cross-memory services

The following list provides an overview and procedure for installing Adabas on an OS IV/F4 system. The rest of this chapter provides details for the steps in the list:

1 Allocate DASD space for the Adabas libraries.

The libraries are restored from the installation tape. Refer to the section **Disk Space Requirements for Libraries** on page 39.

2 Allocate DASD space for the Adabas database.

For better performance, distribute the database files over multiple devices and channels. Refer to the section **Disk Space Requirements for the Database** on page 39.

3 Specify the partition or OS IV/F4 address space for running the Adabas nucleus.

Refer to the section **Adabas Nucleus Address Space Requirements** on page 39.

4 Restore the Adabas libraries from the installation tape.

Use the tape positioning instructions that accompany the tape. Refer to the section **Installing the Release Tape** on page 40.

5 Install the Adabas SVC.

Refer to the section **Initializing the Adabas Communication Environment** on page 41.

Steps 6–15 require changes to the setup definitions (“customizing”) as described under the OS IV/F4 section of **Installing an Adabas Database** on page 47.

6 Allocate and format the Adabas database with the ADAFRM utility job.

7 Define the global database characteristics with the ADADEF utility job.

8 Load the demonstration (demo) files with the ADALODE, ADALODV, and ADALODM jobs.

9 Start the Adabas nucleus and test the Adabas communications with the ADANUC job.

10 If appropriate, test Adabas address space communications by running ADAREP in MULTI mode with the CPEXLIST parameter.

11 If appropriate, load the Adabas Online System (AOS) selectable unit into a Natural system file by running the AOSINPL job. Alternatively, install the AOS demo version delivered with Adabas: see page 139.

12 Terminate the Adabas nucleus with the ADAEND operator command using the MVS Modify command (“F”).

13 Back up the database by running the ADASAV utility job.

14 Insert the ADARUN defaults by running the DEFAULTS job.

15 Install the required TP link routines for Adabas (see the chapter **Installing Adabas with TP Monitors** starting on page 57).

Preparing to Install Adabas

The major steps in preparing for Adabas installation are

- checking for the correct prerequisite system configuration; and
- allocating disk and storage space.

The following sections discuss the nominal disk and storage space requirements and how to allocate the space.

Disk Space Requirements for Libraries

The minimum 3390 disk space requirements for the Adabas libraries as follows:

Libraries	3390 Cylinders	3390 Tracks	Directory Blocks
Load	8	120	40
Source	3	45	20
JCL	1	15	10

Note:
You can isolate user programs from the Adabas load library by creating a separate load library that contains only those modules needed to execute user programs in MULTI-user mode and linked with ADAUSER. For Version 7.4, the modules required by user programs are:

ADAIOR	*ADAMLF*
ADAIOS	*ADAPRF*
ADALNK	*ADARUN*

Disk Space Requirements for the Database

The actual database space needed by Adabas depends on user requirements.

The minimum 3390 disk space requirements for the database are:

Database Component	3390 Cylinders Required	3390 Tracks Required
ASSOR1 (Associator)	20	300
DATAR1 (Data Storage)	60	900
WORKR1 (Work space)	15	225
TEMPR1 (temporary work space)	15	225
SORTR1 (sort work space)	15	225

Adabas Nucleus Address Space Requirements

The typical Adabas nucleus requires at least 800–1024 kilobytes to operate. The size of the nucleus address space may need to be larger, depending on the ADARUN parameter settings. Parameter settings are determined by the user.

Installing the Release Tape

The following job control (JCL) restores the Adabas libraries:

```
//ADAREST   JOB     ...
//STEP1     EXEC    PGM=IEBCOPY
//LOADT     DD      DSN=ADAvrs.LOAD,DISP=(OLD,PASS),
//                  UNIT=TAPE,VOL=SER=vvvvvv,
//                  LABEL=(n,SL)
//SOURCET   DD      DSN=ADAvrs.SRCE,DISP=(OLD,PASS),
//                  UNIT=TAPE,VOL=SER=vvvvvv,
//                  LABEL=(n,SL)
//JCLT      DD      DSN=ADAvrs.JOBS,DISP=(OLD,PASS),
//                  UNIT=TAPE,VOL=SER=vvvvvv,
//                  LABEL=(n,SL)
//*
//*                     ADABAS VERSION v LIBRARIES:
//*
//LOAD      DD      DSN=ADAvrs.LOAD,DISP=(NEW,CATLG),
//                  UNIT=dasdev,VOL=SER=volser,
//                  SPACE=(CYL,(cyl,,blk)),
//                  DCB=(RECFM=U,BLKSIZE=6447)
//SOURCE    DD      DSN=ADAvrs.SRCE,DISP=(NEW,CATLG),
//                  UNIT=dasdev,VOL=SER=volser,
//                  SPACE=(CYL,(cyl,,blk)),
//                  DCB=(RECFM=FB,BLKSIZE=6000,LRECL=80)
//JCL       DD      DSN=ADAvrs.JOBS,DISP=(NEW,CATLG),
//                  UNIT=dasdev,VOL=SER=volser,
//                  SPACE=(CYL,(cyl,,blk)),
//                  DCB=(RECFM=FB,BLKSIZE=6000,LRECL=80)
//SYSPRINT  DD      SYSOUT=X
//SYSIN     DD      *
 COPY  INDD=LOADT,OUTDD=LOAD
 COPY  INDD=SOURCET,OUTDD=SOURCE
 COPY  INDD=JCLT,OUTDD=JCL
/*
//
```

—where

blk	is the number of directory blocks for this library
cyl	is the number of cylinders for this library
vrs	is the Adabas version level
vvvvvv	is the volume serial number for the Adabas installation tape

n	is the position of the dataset on the Adabas installation tape
volser	is the volume/serial number of the disk used for the Adabas library
dasdev	is the DASD device type

Refer to the *Report of Tape Creation* for the correct Adabas library sequence. The DCB statements in the above example are not required on the IEBCOPY step.

Initializing the Adabas Communication Environment

ADASIR must be linked into either SYS1.LINKLIB or into a library concatenated to SYS1.LINKLIB by the LNKLSTxx member of SYS1.PARMLIB. This requires re-IPLing the system to install the Adabas SVC. Note that the CLPA option must first be specified as an IPL-selected option.

Router Installation Overview

1. Restore the Adabas installation tape.

2. Apply the ZAP to define the subsystem (“ADAB”) as described in the section **Executing ADASIR** on page 46.

   If necessary, also do the following operating-system-dependent steps:

OS IV/F4	Apply the ZAP to modify the SVC table entry for a type 3 SVC as described in the section **Setting the SVC Table** on page 42.
MSP/EX	Modify the member KAASVCnn to define the Adabas router under MSP/EX as described in the section **Setting the SVC Table** on page 42.
MSP/EX	Modify the member SUBSYSxx to add the subsystem as described in the section **Executing ADASIR** on page 46.

   Apply the ZAP as described in the section **Setting the ID Table Count and SVC Number** on page 44.

   If necessary, relink the SVC as described in the section **Linking the Adabas SVC** on page 45.

3. Re-IPL the operating system with CLPA to load and execute ADASIR.

Setting the SVC Table

OS IV/F4 systems must define the SVC as a type 3 with no locks.

The VERIFY and REPLACE statement examples in the following sections show how to alter an available SVC entry to define it as type 3.

For OS IV/F4 MSP E20

Use the following statements to alter an available SVC entry to define it as type 3:

```
NAMEX KAANUC01 SVCTABLE
VER offset  xxxx
REP offset  C000          (type 3, no locks)
```

—where

offset	is the offset to byte 4 of the SVC entry for the specified SVC.
xx (or) xxxx	is the existing contents of SVC bytes 4 and 5 at the specified offset.

Note:
The SVC table entry is eight bytes long. The fourth byte of each entry specifies the type attribute, and the fifth byte specifies the lock attribute (see bytes 5 and 6 of the following E20 AE example).

For OS IV/F4 MSP E20 AE

Use the following statements to alter an available SVC entry to define it as type 3:

NAMEX KAANUC01 SVCTABLE
VER offset xxxx xxxx
REP offset **C000 00**yy

—where

offset	is the offset to byte 5 of the SVC entry for the specified SVC.
xxxx xxxx	is the existing contents of SVC bytes 5–8 at the specified offset.
yy	is the value of the selected AMODE option as described below in flag 4 (for example, B’10nn nnnn’ for AMODE=31).

Note:
The SVC table entry is eight bytes long, and has the following format:

Bytes	Meaning	
1 – 4	SVC entry address	
5	Flag byte:	
	Type 1 SVC:	B’0000 nnnn’
	Type 2 SVC:	B’1000 nnnn’
	Type 3 and 4 SVC:	B’1100 nnnn’
	Type 6 SVC:	B’0010 nnnn’
6	Flag 2:	
	Local lock:	B’10nn nnnn’
	CMS lock:	B’11nn nnnn’
7	Flag 3 (reserved)	
8	Flag 4:	
	AMODE=24:	B’00nn nnnn’
	AMODE=31:	B’10nn nnnn’
	AMODE=ANY:	B’01nn nnnn’

For OS IV/F4 MSP 20AE/EX

Either create or modify member KAASVCnn in SYS1.PARMLIB. This member describes the attributes of the SVCs used in the MSP/EX system.

The following characteristics can be used for the Adabas SVC:

- Type 3
- No locks

The entry format is:

NUM=svc**,TYPE=3,EPNAME=**entryname**,AM=31,APF=NO**

—where

svc	is the Adabas SVC number.
entryname	is the Adabas SVC entry name.

Setting the ID Table Count and SVC Number

ADASIR detects and supports either the MSP/20, or MSP/AE and MSP/EX without cross-memory services, based on PTF level 91212 or above running on a FACOM model M760 or higher.

ADASIR does not accept passed parameters. As a result, the ID table entry count, SVC number, and operating mode must be zapped as follows:

```
NAMEX ADASIR
*  ADABAS 7 SVC number
VER 0034 0A00              (this default is invalid)
REP 0034 0Axx              ('xx' is the SVC number)
```

To enter the allowed maximum count of ID table entries, issue the following ZAP:

```
NAMEX ADASIR
*  number of ID Table entries
VER 0020 0000,000A       (default = 10)
REP 0020 0000,00xx       ('xx' is the maximum entry count)
```

To set the operating environment, make the following ZAP;

```
NAMEX ADASIR ADASIR
VER 0024 00              (this default is invalid)
REP 0024 xx              ('xx' is the operating environment, as follows:
                               01:  native mode
                               02:  AVM/EF mode
                               03:  AVM/EX mode)
```

Note:
To get the VMID under AVM/EX, you must set the QAL for the AS option of AVM; otherwise, the VMID is "NO".

Using ADASIR

ADASIR Function for OS IV/F4

ADASIR receives control when the master scheduler is initialized. The operator is prompted for NRIDTES and SVCNR values that were not correctly zapped. Message ADAS03 indicates that ADASIR ended normally, while message ADAS04 indicates an error condition that prevents Adabas from being used until the problem is corrected and the system is re-IPLed. For specific message information, refer to the *Adabas Messages and Codes* manual.

Linking the Adabas SVC

Link the Adabas OS IV/F4 type 3 SVCs into SYS1.LPALIB with the name “JFF00nnp”, where “nn” are the first two digits of the SVC number and “p” is the character equivalent of the last SVC digit (as an example, when the last SVC digit is “1”, “p” must be set to “A”), which must be non-zero.

For MSP 20 AE Systems

For OS IV/F4 MSP 20 AE systems, the following example applies:

```
//A42L31    JOB CLASS=A,MSGCLASS=X
//IEWL2    EXEC PGM=IEWL,
//              PARM='LET,LIST,NCAL,XREF,RENT,SECTION=(31)',REGION=1024K
//SYSPRINT   DD SYSOUT=X
//SYSUT1     DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//ADALIB     DD DSN=user.loadlib,DISP=SHR      <--ADASVC loadlib
//SYSLMOD    DD DSN=SYS1.LPALIB,DISP=SHR
//SYSLIN     DD *
 CHANGE ADASVC(JFF00nnp)
 INCLUDE ADALIB(ADASVC)
 NAME JFF00nnp(R)
/*
```

For Non-MSP 20 AE Systems

The following example shows how to link the SVC for non-MSP 20 AE systems:

```
//      (job card)
//LKED      EXEC  PGM=IEWL,
//          PARM='XREF,LIST,LET,NCAL,RENT,REUS'
//SYSPRINT  DD SYSOUT=X
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD DSN=SYS1.LPALIB,DISP=SHR       <-- Target loadlib
//ADALIB    DD DSN=user.loadlib,DISP=SHR      <-- ADASVC loadlib
//SYSLIN    DD *
 CHANGE ADASVC(JFF00nnp)
 INCLUDE ADALIB(ADASVC)
 NAME JFF00nnp(R)
/*
```

Executing ADASIR

For OS IV/F4 Environments

"ZAP" the ADASIR entry into the subsystem table named KDJJSSNT in SYS1.LINKLIB.

Entries in the subsystem table are 80 bytes long. The table begins directly with the first entry.

To find an available entry for Adabas, dump the module KDJJSSNT in SYS1.LINKLIB and search for an empty 80-byte entry. Empty entries contain all blanks.

Use the following ZAP to add "ADAB" as a valid subsystem:

```
NAMEX KDJJSSNT KDJJSSNT
VER offset 40404040,4040404040404040
REP offset C1C4C1C2,C1C4C1E2C9D94040
```

—where "offset" is the offset to the beginning of the first available subsystem table entry specifying "ADAB,ADASIR", as defined in the REP statement.

Note:
The ADASIR program loads a channel appendage KHH019KU from SYS1.LINKLIB.

At this point, you are ready to IPL the ADASIR routine (with CLPA). Any messages or codes occurring while executing ADASIR are described in the *Adabas Messages and Codes* manual.

For OS IV/F4 MSP (E20, AE and EX) Environments

Define a SUBSYSTEM entry for Adabas by either creating or modifying the member SUBSYSxx in SYS1.PARMLIB. The format is as follows:

SNDSUB SUBNAME=name**,PGM=ADASIR**

—where “name” is the subsystem name (the default is “ADAB”). The value “name” must be the same as the name in the Adabas router, offset X‘28’.

If you already have an “ADAB” subsystem in the system, apply the following ZAP:

```
NAMEX ADASVC
VER 0028 C1C4C1C2
REP 0028 subname
```

—where “subname” is a four-character name other than “ADAB” specified by SUBNAME.

Installing an Adabas Database

Once you have restored the Adabas installation tape and have installed the ADASVC, you can

- migrate an existing Adabas database to the new version; or
- install a new version the Adabas database.

Messages or codes that occur during the installation are described in the *Adabas Messages and Codes* manual; utilities are described in the *Adabas Utilities Manual*.

Migrate an Existing Database

Use the ADACNV utility to migrate existing databases to new releases of Adabas. See the *Adabas Utilities Manual* for more information.

Install a New Database

Step 1. Allocate and format the Adabas database

Customize and run the ADAFRM utility job to allocate and format the Adabas database. The following specific items must be customized (for references to Adabas utility parameters; see the specific utility descriptions in the *Adabas Utilities Manuals*:

- dataset names for the database and libraries;
- volumes for libraries and datasets for the database;
- space allocation for database datasets;
- the Adabas SVC number, the database ID, and database device type(s);
- sizes of the datasets for each ADAFRM statement.

Step 2. Define the global database characteristics

Customize and run the ADADEF utility job to define the global definition of the database. The following items must be customized:

- dataset names of the database and libraries;
- the Adabas SVC number, the database ID, and database device type(s);
- ADADEF utility parameters.

Step 3. Load the demonstration (demo) files

Customize and run the job

- ADALODE to load the sample demo file EMPL;
- ADALODV to load the sample demo file VEHI; and
- ADALODM to load the sample demo file MISC.

The following items must be customized:

- dataset names for the database and libraries;
- the Adabas SVC number, the database ID, and database device type(s);
- ADALOD utility parameters.

Step 4. Start the Adabas nucleus and test the Adabas communications

Customize run the job ADANUC to start up the Adabas nucleus. The following items must be customized:

- dataset names of the database and libraries;
- the Adabas SVC number, the database ID, and device type(s);
- ADANUC parameters.

Step 5. Test Adabas address space communications, if appropriate

Customize and run the job ADAREP in MULTI mode with the CPEXLIST parameter to test Adabas address space communications. The following items must be customized:

- dataset names of the database and libraries;
- the Adabas SVC number, the database ID, and device type(s);
- ADAREP utility parameters.

Step 6. Load the Adabas Online System selectable unit, if appropriate

Customize and run the job AOSINPL to load the Adabas Online System (AOS) into a Natural system file using a batch version of Natural 3.1 or above.

Alternatively, install the AOS demo version delivered with Adabas: see page 139.

Step 7. Terminate the Adabas nucleus

Communicate with the Adabas nucleus to terminate the session either with an ADAEND operator command using the MVS Modify command

F jobname**,ADAEND**

—or

P jobname

—where “jobname” is the started nucleus’ job or task name.

Step 8. Back up the database

Customize and run the ADASAV utility job to back up the version 7 database. The following items must be customized:

- dataset names of the database and libraries;
- the Adabas SVC number, the database ID, and device type(s);
- ADASAV utility parameters.

Step 9. Insert the ADARUN defaults

The member DEFAULTS in the Adabas JCL library can be modified to set the ADARUN defaults. The following items must be customized:

- dataset names for libraries; and
- ADARUN user defaults:
  - device type(s) (default: 3380)
  - SVC number (default: 249)
  - database ID (default: 1)

Customize and run the DEFAULTS job to set the ADARUN defaults using the OS ZAP utility.

Step 10. Install the required TP link routines for Adabas

Refer to the chapter **Installing Adabas with TP Monitors** starting on page 57 for the TP link routine procedure.

SVC Integrity Validation

In the past, the presence of multiple SVCs with the same subsystem ID has resulted in a single ID table being used by different SVCs. This has caused problems, some of them serious (abnormal nucleus termination or corruption of the database).

To eliminate this danger, the version 7 SVC checks the validity by ensuring that the SVC accessing the ID table is the same as the one that was used by ADASIP/ADASIR to initialize the table. If the SVCs are not the same, an ABEND 650 occurs.

ABEND 650 results from a nucleus specifying an incorrect SVC number in its ADARUN parameters. It can occur during Adabas initialization, during the first Adabas call from a user program, or when the ID table is queried by another Software AG server such as Entire Net-Work.

Requirements for Cross-Memory Services

The following hardware/software is required to use cross-memory services with MSP EX:

- FACOM M760, or above;
- MSP EX level PTF91121 operating system, or above.

With cross-memory services, the Adabas command queue and attached buffers are located in the address space instead of the common storage area (CSA), where all components except the ID table are usually located.

When cross-memory services is used, the number of system linkage indexes can be increased from the default of 55 (X'37') by using the following ZAP:

```
NAMEX KAANUC01 KAACMTBL
VER 004E 0037        (default = 55)
REP 004E 0xxx        (maximum: 1024)
```

To get the VMID under AVM/EX, you must set the ASP option of AVM to "ON"; otherwise, the VMID will be null.

Using EXCPVR

CPU usage is reduced considerably by using EXCPVR. This requires that ADARUN be APF-authorized and that all Adabas modules be contained in APF-authorized libraries. When placed in an APF-authorized library, ADARUN allows the Adabas nucleus and utilities to use EXCPVR.

All Adabas modules except the Adalink must run with RMODE=24 (the default); the Adalink can run with any RMODE.

When the EXEC statement specifies PGM=ADARUN, the AMODE is determined as follows:

- The default AMODE=31 is used unless ADARUN was relinked using either the AMODE=24 EXEC statement parameter or the MODE AMODE(24) linkage editor control statement.
- An AMODE of 31 is changed to 24 before affected macros (data management, for example) are run, and then changed back to 31 thereafter.

Adabas performance can be improved by using EXCPVR to improve channel program translation time. For Adabas to invoke EXCPVR automatically, the Adabas modules must be in an APF-authorized library and ADARUN is linked with SETCODE AC(1), the default, as shown in the following example:

```
//LINKRUN    EXEC  PGM=IEWL,PARM='REUS'
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//ADALIB    DD DSN=user.loadlib,DISP=SHR
//SYSLMOD   DD DSN=user.loadlib,DISP=SHR      <==APF-authorized library
//SYSLIN    DD *
  INCLUDE  ADALIB(ADARUN)
  SETCODE  AC(1)
  NAME  ADARUN(R)
```

Creating a Shareable ADARUN

The ADARUN module delivered in the Adabas load library is neither reentrant nor reusable.

A reusable ADARUN permits several programs running in the same address space to share the same ADARUN and ultimately, the same copy of ADALNK. This is important when it is necessary to have only one Adabas user ID for the different programs, and is also needed if single copies of ADALNK user exits are required.

To create a shareable ADARUN

- Relink it with the REUS attribute using the sample job JCLLINRR in the MVSJOBS library.

If both nonreusable and reusable versions of ADARUN are required, they must be located in different load libraries since both must be loadable using the name “ADARUN”.

Storage above 16 MB

Adabas can acquire a number of its required areas, including buffer space, above the 16-MB addressing limit, allowing Adabas to increase the buffer pool size. Storage above 16 MB is available on OS IV/F4 MSP 20 AE and MSP/EX operating systems.

To reverse the space allocation to be below the 16-MB limit, set the SECTION value in the PARM statement in the example below to (24):

```
//LINKRUN    EXEC  PGM=IEWL,
                 PARM='LET,LIST,NCAL,XREF,RENT,SECTION=(24)'
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//ADALIB    DD DSN=user.loadlib,DISP=SHR
//SYSLMOD   DD DSN=user.loadlib,DISP=SHR      <==APF-authorized library
//SYSLIN    DD *
  INCLUDE ADALIB(ADARUN)
  SETCODE AC(1)
  NAME ADARUN(R)
```

In addition, Adabas must be run with a sufficient REGION specification, either on the JOB or EXEC statement or as an installation default. For example:

//BIG JOB ...,REGION=30M,...

The attached buffers and the Adabas command queue can also be allocated in storage above the 16-MB line. These components are allocated in ECSA if the SVC table entry “FLAG 4” specifies AMODE=31.

Adalink Considerations

Note:
For information about setting up a database for data conversion using the universal encoding service (UES), see the chapter ***Connecting UES-Enabled Databases*** *starting on page 107.*

User Exit B (Pre-command) and User Exit A (Post-command)

One or two user exits may be linked with an Adalink routine:

- UEXITB receives control **before** a command is passed to a target with the router 04 call.

  Note:
  Special commands emanating from utilities and from Adabas Online System are marked as "physical" calls. These calls must be bypassed in user exits. These calls have X'04' in the first byte (TYPE field) of the command's Adabas control block (ACB). UEXITB must check this byte and return if it is set to X'04'. Be sure to reset R15 to zero on return.

- UEXITA receives control **after** a command has been completely processed by a target, the router, or by the Adalink itself.

  At entry to the exit(s), the registers contain the following:

Reg.	Content
1	Address of the UB If the flag bit UBFINUB is reset, the contents of the halfword at ADABAS + X'86' have been moved to UBLUINFO. If those contents are greater than zero, the two bytes starting at UBINFO (UB+X'40') have been set to zero. If UBFINUB is set, no changes can be made to the UB or ACB (except for ACBRSP).
2	Address of a 16-word save area (for ADALNC only)
13	Address of an 18-word save area (for non-CICS Adalink exits)
14	Return address
15	Entry point address: UEXITB or UEXITA

Any registers except register 15 that are modified by the user exits must be saved and restored; the address of a save area for this purpose is in register 13 (or register 2 for ADALNC).

If at return from UEXITB register 15 contains a value other than zero (0), the command is not sent to the target but is returned to the caller. The user exit should have set ACBRSP to a non-zero value to indicate to the calling program that it has suppressed the command: response code 216 is reserved for this purpose.

The UEXITB exit may set the UB field UBLUINFO to any lesser value, including zero; an ABEND occurs if the user exit sets UBLUINFO to a greater value. The UBLUINFO length cannot be changed when any other exit is used; for example, Review or Fastpath.

The user information received by a UEXITA exit may have been modified; this modification may include decreasing its length, possibly to zero, by any of the Adalink user exits.

An Adalink routine can return the following non-zero response codes in ACBRSP:

213 No ID table
216 UEXITB suppressed the command
218 No UB available

At least the following three EQUates, described at the beginning of the source, can be modified before an Adalink routine is assembled. In some Adalink routines, however, the corresponding information can be zapped:

LOGID is the default logical ID, ranging in value from 1 to 65535. The default is 1.
LNUINFO is the length of the user information to be passed to Adalink user exits, ranging in value from 0 to 32767. The default is 0.
SVCNR is the Adabas SVC number; its range of values and the default depend on the operating system.

The first 152 (X'98') bytes of all Adabas Adalinks must maintain the following structure:

Offset	Label	Contents	Meaning
00	ADABAS		Entry code
12		CL6'ADALNx'	Program name
18		XL4'yyyymmdd'	Assembly date
1C		A(ZAPTAB)	Address of ZAP table
20	PATCH	XL96'00'	Patch area
80	LNKLOGID	AL2(LOGID)	Default logical ID (default: 1)
82		XL2'00'	Reserved
84	LNKSVC	SVC SVCNR	Executable SVC instruction for Adabas SVC (default: operating-system-dependent)
86	LUINFO	Y(LNUINFO)	Length of user information (default: 0)
88	VUEXITA	V(UEXITA)	Address of user exit after call (weak)
8C	VUEXITB	V(UEXITB)	Address of user exit before call (weak)
90	ADABAS51	CL8'ADABAS51'	IDT ID

ADAUSER Considerations

ADAUSER is a program that links the user to Adabas. It is specific to an operating system and is independent of release level and mode. It can be used in batch and in some TP environments.

ADAUSER operates in the following way:

- ADAUSER contains the entry point ADABAS and should be linked with all user programs that call Adabas. No other programs containing the CSECT or entry point name “ADABAS” can be linked in these load modules/phases.
- On the first call to Adabas, ADAUSER loads the latest version of ADARUN. This makes the calling process release-independent. Subsequent Adabas calls bypass ADARUN.
- ADARUN processes its control statements. If the ADARUN PROGRAM parameter has the (default) value USER, ADARUN loads ADAIOR and, depending on whether the ADARUN MODE parameter specifies MULTI or SINGLE, loads the appropriate TP Adalink or ADANUC, respectively. This makes the calling process mode-independent.

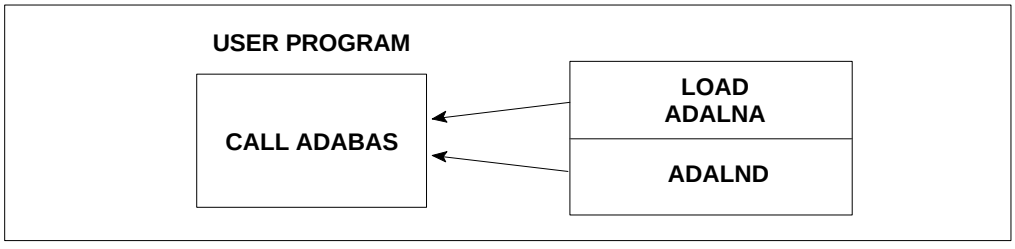
4

INSTALLING ADABAS WITH TP MONITORS

This chapter provides information needed to install Adabas with teleprocessing (TP) monitors: Com-plete, CICS, IMS, Shadow, TSO; AIM/DC; AITM, TIAM and UTM. Information about using Adabas with TP monitors is contained in other chapters as well, particularly in the chapters describing Adabas installation by operating system.

Installing Adabas with AIM/DC

This section describes installation of the Fujitsu (FACOM) AIM/DC TP monitor with Adabas.



The job AIMASM is provided for assembling the source members ADALNA and ADALND, the AIM/DC-dependent link routines. You must first customize the JCL for AIMASM to select the MVS operating system MACLIB (SYS1.MACLIB) containing the LOAD macro.

Before AIMASM can be run, ADALNA must be customized to select the following options:

Option	Value	Specify . . .
SVCNR	249 | nn	the value of the MSP SVC number.
LOGID	1 | nn	the value of the default logical database ID ranging 1-255.
NUBS	50 | nn	the value for the number of UBs (user blocks) to be created by ADALNA. This value must be high enough to handle the maximum possible number of concurrent Adabas requests.

Note:
The modules ADALNA and ADALND are linked together to form ADALNA.

Preparing Adabas Link Routines for IBM Platforms

Use the High-Level Assembler

The IBM high-level Assembler is **required** when assembling the Adabas link routines for TP monitors. The high-level Assembler generates 4-digit year assembly dates into the load modules using the &SYSDATC assembly variable. The older Assemblers H and F do **not** support 4-digit year assembly dates.

It is possible to assemble the Adabas link routines without the high-level Assembler, either as is (ignore the assembly error), or with the &SYSDATC variable changed to “yyyymmdd”, any valid 4-byte unsigned decimal assembly date where “yyyy” is a 4-digit year, “mm” is a 2-digit month, and “dd” is a 2-digit day. The assembly date field is restricted to 4 bytes in the load module.

Addressing Mode Assembly Directives

The Adabas link routines now have AMODE and RMODE assembly directives in the source. These allow the linkage editor to produce warning messages when conflicting AMODE or RMODE linkage-editor control statements are encountered in the link JCL, JCS, or EXECs.

These assembly directives also serve to document the preferred AMODE and RMODE for each link routine. It is important to note that in and of themselves, these directives do not alter the actual addressing mode of the link routine during execution.

The batch/TSO link routine ADALNK has the following AMODE and RMODE assembly directives:

```
ADABAS AMODE 31
ADABAS RMODE 24
```

For the CICS and IMS/ESA link routines (modules LNKOLM, LNKOLSC, and ADALNI) the directives are

```
ADABAS AMODE 31
ADABAS RMODE ANY
```

Modifying the Assembly Directives

These directives may be changed by modifying the source members before assembling them, or they may be overridden by linkage editor control statements. For example, to link the batch/TSO ADALNK module with AMODE 31 and an RMODE of ANY, the following control statements may be provided as input to the linkage editor:

```
MODE AMODE(31),RMODE(ANY)
ENTRY ADABAS
NAME ADALNK(R)
```

The linkage editor control statements override the Assembler directives in the source module.

Note:
Future releases of the Adabas link routines may require an AMODE of 31 and an RMODE of ANY to function properly in the OS/390 or z/OS environment. Software AG strongly recommends that you evaluate application programs and update them to conform to this standard.

For more information about the AMODE and RMODE directives and their effects on the assembler, linkage editor, and execution, consult the IBM *MVS/ESA Extended Addressability Guide*.

UES-Enabled Link Routines

For Adabas version 7.4, UES is enabled by default for the batch/TSO, Com-plete, and IMS link routines. It is **not** necessary to disable UES support. Applications that do not require UES translation continue to work properly even when the UES components are linked with the Adabas link routines. See the chapter **Connecting UES-Enabled Databases** starting on page 107 for more information.

Disabling UES Support

However, if for some reason you feel it necessary to disable UES support in the Adabas link routines, use the following procedure to do so:

1. Edit the source member ADALCO, ADALNI, ADALNK, or ADALNKR. Set the &UES Boolean assembler variable to zero (0) by commenting out the source line where it is set to 1 and removing the comment from the line where it is set to 0.

2. Assemble the link routine after making any other necessary modifications to the equates and other directives in the source module as required by your installation.

3. Link the Adabas link routine and do **not** include any of the UES components (that is, LNKUES, ASC2EBC, or EBC2ASC).

Installing Adabas with CICS

CICS/ESA 3.2 and above for OS/390 or z/OS environments **must** run a current version of Adabas and use the command-level link component. The macro-level link routine ADALNC is no longer supported in this environment.

The Adabas command-level link routine supports the CICS transaction server (CTS) environment.

Notes:

1. *The OPID option for the USERID field is not supported under CICS/ESA 3.2 and above; therefore, it is not provided with the command-level link routine.*
2. *When running under CICS 4.1, the CICS components from Adabas 5.3.3 or above are required.*

The following sections describe specific points of Adabas/CICS installation and operation from the CICS perspective.

Adabas Bridge for VSAM Considerations

If you are running Adabas Bridge for VSAM 4.2 or 5.1 under CICS, you **must** run CICS 3.3 or above and the Adabas version 7.1 or above command-level link routine.

Note:
Adabas Bridge for VSAM version 4.1.1 must use the Adabas command-level link routine included in the AVB 4.1.1 source library.

CICS MRO Environment Requirements

If you run the Adabas CICS command-level link routine with the CICS multiple region option (MRO), you must set the ADAGSET option MRO=YES and use the default value for the ADAGSET NETOPT option (see page 71).

You can use the ADAGSET NTGPID option to provide a 4-byte literal for the Adabas communication ID to be used by the Adabas SVC when applications that call Adabas span multiple application regions.

Alternatively, you can create a user exit B (UEXITB) for the link routine that

- sets UBFLAG1 (byte X'29' in the UB DSECT) to a value of X'08' (UBF1IMSR); and
- places a 4-byte alphanumeric value in the UB field UBIMSID.

The exit then allows the Adabas SVC to provide a proper Adabas communication ID in the Adabas command queue element (CQE) even when transactions originate in multiple regions.

Using CICS Storage Protection

The storage protection mechanism (STGPROT) was introduced under CICS/ESA 3.3. Storage protection permits resources to access either CICS or user storage by using the OS/390 storage protection keys. Resources defined to operate in

- user key may not overwrite CICS storage, thus affording a degree of protection to CICS.
- CICS key may read or write either CICS or user key storage, affording the highest degree of access to CICS resources.

To use storage protection with Adabas, you must either

- use the task-related user exit (ADAGSET TRUE=YES); or
- define the Adabas link routine with EXECKEY(CICS) in RDO.

Software AG recommends using the task-related user exit.

Standard Versus Enhanced Installation

All supported versions of the command-level link routine can be installed using the “standard” installation, which comprises steps 1 through 3 of the installation procedure beginning on page 76.

Steps 4 and 5 are required in order to use the enhanced features of the command-level link routine:

- CICS transaction isolation;
- ADASAF under CICS 4.1 or above; and
- an operationally reentrant command-level link after initialization.

Step 6 is used to install the optional DISPGWA program. The DISPGWA program is only available with the enhanced installation.

The enhanced installation is **required** if

- Adabas SAF Security (ADASAF) is being used under CICS 4.1 or above;
- CICS transaction isolation is used;
- Adabas Bridge for VSAM is used (version 4.1 uses a separate command-level link routine included in the AVB version 4.1 source library; versions 4.2 and 5.1 use the same routine as Adabas); or
- the DISPGWA storage display program is used.

CICS Transaction Isolation

The enhanced Adabas CICS command-level link components take advantage of the transaction isolation facility provided by CICS/ESA 4.1 when running with specific hardware under OS/390 or z/OS.

Transaction isolation is an extension of the storage protection mechanism (see page 62). It further protects CICS resources by isolating them in “subspaces”. This protects user key resources from one another, and protects CICS key resources from the CICS kernel.

Transaction isolation can be enabled globally through the TRANISO system initialization (SIT) parameter, and for each CICS transaction with the new resource definition ISOLATE keyword.

Transaction isolation places some restrictions on CICS resources that must be available both during the life of the CICS system and to all transactions running in the CICS system.

The Adabas CICS command-level link components must be defined to CICS with the proper storage access to ensure proper operation when transaction isolation is active:

- The Adabas CICS command-level link routine, comprising the LNKOLSC, LNKOLM, and CICS entry and exit code, must be defined to CICS as a “user key” program.
- The LNKTRUE and LNKENAB (ADATRUE and ADAENAB) programs must both be defined as “CICS key” programs.

This permits the correct degree of isolation between applications invoking the link routine and the Adabas CICS task-related user exit (TRUE), which interacts directly with CICS resources.

When CICS transaction isolation is active, user SVCs cannot execute from the CICS region. SVCs can be executed in CICS key, however, during the PLT phase of the CICS startup and termination. For this reason, the module LNKENAB is provided to execute during the PLTPI phase of CICS initialization.

ADASAF and CICS 4.1

The Adabas SAF Security (ADASAF) module resides in the Adabas nucleus region to handle the authorization of Adabas resources.

Under CICS 4.1, this module depends on the external security identifier (user sign-on) being extracted from the access control environment element (ACEE) in the caller’s address space, and being passed on to Adabas by the SVC.

Under CICS 4.1, the command-level link routine must utilize a CICS task-related user exit to extract the external security identifier from the ACEE. The Adabas task-related user exit in the module LNKTRUE has been included in Adabas for this purpose.

Note:
The Adabas task-related user exit LNKTRUE ***is not required*** *for ADASAF under CICS versions prior to 4.1.*

Operationally Reentrant Link Routine after Initialization

The Adabas command-level link routine is operationally reentrant and not self-modifying after initialization. During execution, a CICS global work area (GWA) is obtained and passed to the command-level link, which uses the GWA to store addresses. This feature is available for CICS version 3.2 and above.

Note:
*Do **not** use the JCL parameter RENT when assembling/linking the command-level link routine.*

DISPGWA Module : Displaying the CICS Global Work Areas

The DISPGWA module is a program provided by Software AG as an optional feature for CICS version 3.3 and above.

The DISPGWA program displays the global work area (GWA) used by the various command-level link components. It can be used to display other areas of CICS that are important to the Adabas command-level link routine when it is executing as a task-related user exit (TRUE). With the help of Software AG personnel, you can use this program to interrogate important data areas during problem determination.

The DISPGWA module is used only if the LNKTRUE module is used, since it is the LNKTRUE module that actually EXTRACTs the global work area.

LNKENAB and LNKTRUE Modules

LNKENAB Module

The LNKENAB module

- starts and enables the task-related user exit LNKTRUE (see the next section) during PLTPI processing.
- defines the length of storage that CICS gives to LNKTRUE as each task is invoked for the first time. The storage remains in CICS until the task terminates and is used by LNKTRUE as a task work area.
- issues an Adabas command to the default target and Adabas SVC defined in the ADAGSET macro in LNKOLSC. The target need not be active.

  The purpose of the command is to derive the address of the IDTH from the first SVC call. All other SVC calls from the command-level link routine are then made using a branch entry into the Adabas SVC.

LNKTRUE Module

When started and enabled by LNKENAB during PLTPI processing, the task-related user exit LNKTRUE module

- permits the command-level link routine to obtain the pointer to the access control environment element (ACEE) in a CICS/ESA 4.1 environment;
- facilitates processing when CICS transaction isolation is installed and enabled; and
- coordinates Adabas transactions through the CICS Resource Manager Interface (RMI) when the Adabas Transaction Manager (ATM) is installed and enabled.

Most of the command-level link components execute under the umbrella of LNKTRUE. This means that any ABEND condition during the execution of LNKTRUE is serious; CICS may even respond by terminating the entire CICS region.

Items that may cause this condition include, but are not limited to

- invalid application parameter lists for Adabas calls;
- inconsistent or incorrect keyword values coded in the ADAGSET macro for the various command-level components;
- user modification to the command-level link components or the task-related user exit; or
- incorrect coding in UEXITA and/or UEXITB components.

Software AG strongly recommends that you test application programs in a CICS region that supports the **non**-task-related user exit version (standard installation) of the command-level link before migrating to a task-related user exit version (enhanced installation) of the command-level link routine.

JCL and Source Members

The JCL members use the sources indicated in the following table:

JCL	Source	Source Description
CICCASM	LNKOLSC	Adabas command-level link routine (dependent portion).
	LNKOLM	Adabas command-level link routine (independent portion).
CICTASM	LNKTRUE	Adabas task-related user exit.
CICEASM	LNKENAB	Adabas PLT-enable program.
CICDASM	DISPGWA	Display program for Adabas global work area (GWA).

Sample Resource Definitions (SMA-jobnumber I005)

Under CICS/ESA 3.3 and above, the preferred method for defining and installing CICS programs and transactions is RDO (resource definition online). The CICS documentation no longer recommends the assembly of PPT and PCT entries to define resources.

The following table provides sample RDO definitions for the Adabas CICS command-level link components. The data has been extracted directly from the CICS CSD file and should be used as a guide for providing comparable information on the CEDA panels.

```
********************************************************************
*   Sample DEFINE control statements for the DFHCSDUP utility.
*   For Adabas V7.4 CICS command-level link routine components.
*
*   These control statements can be used as input to the DFHCSDUP
*   CICS CSD update utility to define the Adabas CICS command-level
*   link routine components on a CICS/ESA system.
********************************************************************
DEFINE PROGRAM(ADABAS) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s COMMAND LEVEL LINK ROUTINE)
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(USER) EXECUTIONSET(FULLAPI)
DEFINE PROGRAM(ADAENAB) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s PLTPI ENABLE ADATRUE PROGRAM)
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(CICS) EXECUTIONSET(FULLAPI)
DEFINE PROGRAM(ADATEST) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s DISPLAY GWA PROGRAM - DISPGWA)
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(CICS) EXECUTIONSET(FULLAPI)
DEFINE PROGRAM(ADATRUE) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s TASK RELATED USER EXIT)
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(CICS) EXECUTIONSET(FULLAPI)
DEFINE TRANSACTION(DGWA) GROUP(ADABAS)
DESCRIPTION(TRANSACTION TO DISPLAY ADABAS GWA)
       PROGRAM(ADATEST) TWASIZE(128) PROFILE(DFHCICST) STATUS(ENABLED)
       TASKDATALOC(ANY) TASKDATAKEY(CICS) STORAGECLEAR(NO)
       RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
       PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO) INDOUBT(BACKOUT)
       RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
       RESSEC(NO) CMDSEC(NO)
```

—where “s” is the system maintenance level of Adabas.

These sample DEFINE statements are located in member DEFADAC in the Adabas version 7.4 CICS command-level source library. They can be modified and used as input to the IBM DFHCSDUP utility to define the Adabas CICS command-level components. Consult the appropriate IBM CICS documentation for information on the DFHCSDUP utility.

Modifying Source Member Defaults (ADAGSET Macro)

The ADAGSET macro is used to create default settings for the command-level link components. This macro exists in each of the installation source members. The macro settings must be identical in every installation source member used.

To facilitate the assembly of the Adabas command-level link routine components, Software AG recommends that you program the ADAGSET macro with site-specific default values and put it in a source library that is available in the SYSLIB concatenation during assembly.

It is critical that the values for the following keywords agree for all components of the Adabas CICS command-level link routine: LOGID, SVCNO, LUINFO, LRINFO, LUSAVE, NUBS, ENTPT, TRUENAM, and ENABNAM.

Step 1 of the installation procedure (page 76) identifies the source members that must be edited for standard and enhanced installation.

The ADAGSET options with their default values (underlined) are described in the following paragraphs.

AVB={ NO | YES }

If Software AG’s Adabas Bridge for VSAM is to be supported by this command-level link routine, specify YES; otherwise, NO (the default).

ENABNM={ ’ADAENAB’ | ’name’ }

Specify the entry point name for the program that is run to enable the Adabas TRUE during CICS PLTPI processing. The value must be a valid program name that matches the module name specified in the DFHPLT table at your site. The default value is ADAENAB. This parameter is ignored if TRUE=NO.

ENTPT={ 'ADABAS' | 'name' }

The name given to the Adabas CICS command-level link routine, which is the combination of LNKOLSC, LNKOLM, and the CICS entry and exit code. This name is used in EXEC CICS LINK commands to invoke Adabas services from CICS application programs. See also notes 1 and 2 in the installation procedure on page 77.

SAF={NO | YES}

Specify whether the Adabas SAF Security (ADASAF) is used. If you are using ADASAF, you must set SAF=YES.

ADASAF require the Adabas task-related user exit (TRUE) when running under CICS/ESA 4.1 or above. When SAF=YES and TRUE=YES, the task-related user exit passes the user's external security ID (sign-on) to Adabas.

If TRUE=YES is not specified in this case, the ADAGSET macro terminates the LNKOLSC, LNKTRUE, or LNKENAB assembly process with an MNOTE and a return code of 16.

TRUE is not required when running ADASAF under CICS/ESA 3.3 or below. The combination SAF=YES and TRUE=NO is valid in such cases.

LADAFP={ 0 | nn}

Specify the length of the work area provided to the Adabas Fastpath exit.

Values from "0" (the default) to 32767 may be specified. Zero ("0") indicates that Adabas Fastpath is not linked with the Adabas command-level link routine. A nonzero value requires that the parameter TRUE=YES is also set and the Adabas task-related user exit (TRUE) is used. Consult the Adabas Fastpath documentation for recommended values.

Note:
This parameter is not yet fully implemented. It is provided for future use by Adabas Fastpath.

LOGID= nnn

Specify the value of the default logical database ID; valid ID numbers are 1-65535.

LRINFO={ 0 | 256}

Specify the length (in bytes) of the Adabas Review data area to be used by the REVEXITB program. The default is zero (Adabas Review is not being used). The minimum (and recommended) value is 256, the size Adabas Review expects when the REVEXITB program is invoked. See the appropriate Adabas Review manual for more information.

LUINFO={ 0 | length}

Specify the length of the user data to be passed from the CICS link routine to Adabas UEXITA and UEXITB. If not specified, the default is zero (no user save area is passed).

LUSAVE={ 0 | size}

Specify the size of the user save area to be used by Adabas UEXITA and UEXITB. If given, a value of 72 or higher must be specified. If not specified, the default is zero (no user data is passed).

LXITAA={ 0 | nn}

Specify the length of the work area provided to the UEXITA program.

Values from “0” (the default) to 32767 may be specified. Zero (“0”) indicates that no UEXITA program is linked with the Adabas command-level link routine and no data is passed to UEXITA.

Note:
This parameter is not yet fully implemented. It is provided for future use by the CICS user exit A program linked with LNKOLM.

LXITBA={ 0 | nn}

Specify the length of the work area provided to the UEXITB program.

Values from “0” (the default) to 32767 may be specified. Zero (“0”) indicates that no UEXITB program is linked with the Adabas command-level link routine.

Note:
This parameter is not yet fully implemented. It is provided for future use by the CICS user exit B program linked with LNKOLM.

MRO={ NO | YES }

If you run the CICS command-level link with the CICS multiple region option (MRO), set MRO=YES; otherwise, use the default value NO. If MRO=YES, NETOPT must be set to NO (the default) to prevent nonunique LU names from multiple application regions. If NETOPT=YES and MRO=YES, an assembler MNOTE and a return code of 16 are produced from the assembly step.

NETOPT={ NO | YES }

Specify YES for the 8-byte user ID to be constructed from the VTAM LU name. Otherwise, the user ID is created from the constant “CICS” plus the four-byte CICS terminal ID (TCTTETI) for terminal tasks. For nonterminal tasks, the user ID comprises the constant “CIC” plus the CICS task number.

If you run with the CICS multiple region option (MRO), you must use the default value for this option. If NETOPT=YES and MRO=YES, an assembler MNOTE and a return code of 16 are produced from the assembly step.

NTGPID=4-byte-value

Specify a 4-byte Natural group ID as required for unique Adabas user ID generation in the CICSplex environment with Natural version 2.2.8 and above. The value is associated with all users who call the Adabas command-level link routine assembled with the specified value.

There is no default value. If no value is specified, the Adabas internal user ID is built in the conventional manner.

Any 4-byte alphanumeric value may be specified, but it must be unique for each Adabas command-level link routine running in a CICSplex, z/OS, or OS/390 image. If more than one NTGPID is required (for example, both test and production Natural 2.2.8), more than one Adabas command-level link routine with associated TRUE must be generated.

If you run with the CICS multiple region option (MRO), you may use NTGPID to provide a 4-byte literal for the Adabas communication ID to be used by the Adabas SVC when multiple application regions call Adabas. See page 62 for more information.

NUBS={ 50 | blocks }

Specify the number of user blocks (UBs) to be created by the CICS link routine. The number of blocks must be large enough to handle the maximum possible number of concurrent Adabas requests.

Note:
The Adabas 6.2 and above command-level link routine obtains storage for the user blocks (the UB pool) above the 16-megabyte line.

PARMTYP={ ALL | COM | TWA }

Specify the area to contain the Adabas parameter list. TWA picks up the parameter list in the first six fullwords of the transaction work area (TWA). COM picks up the list in the COMMAREA, followed by the normal Adabas parameter list. The COMMAREA list must be at least 32 bytes long and begin with the label “ADABAS52”. ALL (the default) uses both the COMMAREA and TWA to pass the Adabas parameters; in this case, the COMMAREA is checked first. PARMTYP=ALL or PARMTYP=COM must be used if the TRUE=YES option is specified.

PURGE={ NO | YES }

The PURGE parameter is used when assembling with CICS 3.2 or above. If YES is specified, the CICS WAIT EXTERNAL will contain PURGEABLE as one of its parameters, allowing the transaction to be purged by CICS if the DTIMOUT value is exceeded and SPURGE is specified. If NO (the default) is specified, the NONPURGEABLE option is generated.

RMI={ NO | YES }

Specify YES if the Adabas task-related user exit (TRUE) is to execute as a resource manager (RM) using the CICS Resource Manager Interface (RMI). RMI=YES is valid only when the Adabas Transaction Manager (ATM) is installed, enabled, and available to users executing in the CICS environment. Consult the *Adabas Transaction Manager Manual* for additional instructions related to the installation of the Adabas TRUE.

SAP={ NO | YES }

Specify YES if Adabas support is required for the SAP application system. If you do not use the SAP application system, enter SAP=NO (the default).

If you specify YES, the LNKOLSC program will detect a SAP initialization call and set the user ID for SAP applications from the constant provided on the initialization call, plus the field ACBADD2. For more information, refer to the supplementary information provided to customers using the SAP application system.

SVCNO={ 0 | nnn}

Specify the value of the Adabas SVC number.

TRUE={ NO | YES }

Specify YES if LNKOLSC will use the Adabas task-related user exit LNKTRUE. If YES is specified, PARMTYP={ALL | COM} and TRUENM='name' must also be specified.

TRUE=YES is required if you are running ADASAF under CICS 4.1 or above. See the ESI parameter on page 69 for more information.

TRUENM= 'name'

Specify the name of the task-related user exit. This parameter is required if TRUE=YES is specified. See also notes 1 and 2 in the installation procedure on page 77.

UBPLOC= {ABOVE | BELOW}

Specify whether the user block (UB) pool is to be obtained above (the default) or below the 16-megabyte line in CICS. The ECB used by the EXEC CICS WAIT WAITCICS or the EXEC CICS WAIT EXTERNAL is included in the UB pool. The "below" setting supports versions of CICS that do not allow ECBs above the 16-megabyte line; that is, CICS/ESA 3.2 or below. Refer to the IBM manual *CICS/ESA Application Programming Reference* for more information.

XWAIT={ NO | YES }

Note:
With Adabas version 6, the default for the XWAIT parameter changed from NO to YES to conform with IBM usage.

Specify whether a standard EXEC CICS WAITCICS (XWAIT=NO) or a WAIT EVENTS EXTERNAL (XWAIT=YES) will be generated into the command-level link component by the assembler process in the LNKOLSC module. YES is the default.

Notes:

1. *If XWAIT=NO is specified, the Adabas 6.2 and above LNKOLSC module issues an EXEC CICS WAITCICS command instead of the EXEC CICS WAIT EVENT command used in previous versions. This conforms with recommended IBM usage of the WAIT and ECB lists in a high-transaction volume CICS system with CICS/ESA version 4.1 and above.*
2. *All EXEC CICS commands are processed by the CICS preprocessor; the ADAGSET parameters cause the subsequent assembly step to "skip" some of the statements.*

   The CICS WAIT EVENTS EXTERNAL (XWAIT=YES) is the recommended interface for CICS/ESA 3.3 and above.

   The CICS WAITCICS statement (XWAIT=NO) is provided for use with CICS/MVS 2.1.2 and for CICS/VSE 2.1 through 2.3. It may also be used for CICS/ESA 3.3 and above, but may result in poor CICS transaction performance or unpredictable transaction results in busy CICS/ESA environments.

   Note:
   If XWAIT=NO is specified for use under CICS/ESA 3.3, IBM APAR PN39579 must be applied to the CICS/ESA 3.3 system. For CICS/ESA 4.1 and above, this APAR is not required.

- **XWAIT Posting Mechanisms**

  CICS WAITCICS (XWAIT=NO) can support a "soft post" of the specified ECB. This has the disadvantage of becoming a low priority dispatchable unit of work in a CICS/ESA environment, since the "hand postable" work is not processed by CICS on every work cycle.

  EXEC CICS WAIT EXTERNAL (XWAIT=YES), on the other hand, allows CICS to make use of its special post exit code, and will always be checked and processed (if posted) on every CICS work cycle.

  For more details on the differences between the various CICS WAIT commands and their relationship to hard and soft posting mechanisms, consult the IBM *CICS/ESA Application Programming Reference Guide* and the texts accompanying IBM APAR PN39579 or "Item RTA000043874" on the IBM InfoLink service.

- **XWAIT and the Adabas SVC / Router**

  The Adabas 6.2 and above SVC is fully compatible with the XWAIT=YES setting. The SVC performs the necessary "hard post" for Adabas callers under CICS/ESA using the Adabas 6 command-level link routine. The same SVC performs a "soft post" for batch callers where the hard post is not required.

If XWAIT=YES is specified and the Adabas SVC is below the 6.2 level, a ZAP is required to provide the “hard post” code preferred for CICS/ESA users:

ZAP AO33024	for an Adabas 5.3.3 SVC
ZAP AO13016	for an Adabas 6.1.2 or 6.1.3 SVC

This ZAP is available from your Software AG technical support representative.

Software AG strongly recommends that you use the Adabas 6.2 or above SVC/router with XWAIT=YES. The ZAPs to earlier SVC/routers may degrade performance for non-CICS Adabas transactions that use the modified SVC/router.

Installation Procedure

Step 1. Modify the ADAGSET Macro for the Source Member(s)

Modify the ADAGSET macro for the source members to be used.

See the section **Modifying Source Member Defaults (ADAGSET)** starting on page 68 for details. Software AG recommends that you modify a common version of ADAGSET and place it in a library available in the SYSLIB concatenation when the Adabas command-level link components are assembled.

Note:
*It is no longer necessary to modify the equates inside the LNKOLSC code. Instead, use the ADAGSET macro to set default values **before** assembly, thus making the process easier and more self-documenting.*

For the standard installation (without the enhanced functions)

Modify the source member ADAGSET to set the following options:

- database ID (LOGID);
- Adabas SVC number (SVCNO);
- any additional options necessary for your site.

For the enhanced installation

Modify the source member ADAGSET to set the following options:

- database ID (LOGID);
- Adabas SVC number (SVCNO);
- the command-level link routine name (ENTPT);
- the task-related user exit name (TRUENM);
- any additional options necessary for your site.

Notes:

1. ***It is critically important that the ENTPT and TRUENM parameters coded in the LNKENAB, LNKTRUE, and LNKOLSC modules are identical.*** *These module names are used to identify the global work area and task-related user exit (TRUE) storage provided for the CICS transaction using the link-specific link component.*
2. *If you are installing multiple instances of the command-level link routine, the ENTPT and TRUENM names must be unique, as there is a one-for-one relationship between a command-level link routine, its associated task-related user exit (TRUE), and the enabling program LNKENAB run at CICS startup.*

Step 2. Modify the JCL Members

Use your editor to modify the JCL members necessary for your installation, and set the library names according to your installation's specifications. The Adabas job library contains the following JCL members:

JCL Member	Installation Type	Required/ Optional	Description
CICCASM	standard and enhanced	required	Adabas command-level link routine modules.
CICTASM	enhanced	required	Adabas task-related user exit
CICEASM	enhanced	required	Adabas PLT-enable program
CICDASM	enhanced	optional	Adabas global work area (GWA) display program.

Make mass changes to the JCL members in the following order:

Order	Change ...	to ...
1	ADABAS.V7nn.COML	Adabas CICS ... source library.
2	ADABAS.V7nn	Adabas V7 prefix.
3	MAC5='CICS'	CICS system prefix.
4	CLIB='CICS'	CICS system prefix for CICS LOAD.
5	RPLLIB='ADABAS'	COMLEV CICS RPL lib.

Step 3. Install the Adabas Command-Level Link Component (SMA-jobnumber I070)

1. For CICCASM, concatenate the Adabas CICS 7.4 source library on the assembler SYSLIB statement in front of the Adabas source library for the version of Adabas you are running.

2. Execute CICCASM.

   This job preprocesses, assembles, and links the Adabas CICS command-level link routine modules LNKOLSC and LNKOLM into a staging load library.

   The final step of the CICCASM job links the LNKOLSC and LNKOLM modules together with CICS entry and exit code to form the Adabas command-level link routine, which is placed in a CICS RPL library.

3. Use DFHCSDUP or the CEDA RDO entry panels to add the following definition to your CICS CSD file:

```
DEFINE PROGRAM(ADABAS) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s COMMAND LEVEL LINK ROUTINE)
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(USER) EXECUTIONSET(FULLAPI)
```

   —where "s" is the system maintenance level of Adabas.

   The Adabas command-level link routine is now installed. This completes the "standard" installation. To install the enhanced functions**,** continue with step 4.

Step 4. Install the Adabas Task-Related User Exit (SMA-jobnumber I070)

The Adabas CICS components reside in the Adabas CICS source library (ACI74s.MVSSRCE).

1. Execute CICTASM.

   This module preprocesses, assembles, and links the Adabas task-related user exit into a staging load library, and then links it with CICS entry and exit code into a CICS RPL library.

   Unless you are following the installation procedure required for running with the Adabas Transaction Manager, expect the unresolved external references TCISYNC and TCIRESYN.

2. Use DFHCSDUP or the CEDA RDO entry panels to add the following definition to your CICS CSD file:

```
DEFINE PROGRAM(ADATRUE) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s TASK RELATED USER EXIT)
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(CICS) EXECUTIONSET(FULLAPI)
```

   —where “s” is the system maintenance level of Adabas.

Step 5. Install the Adabas PLT-Enable Program (SMA-jobnumber I070)

The Adabas CICS components reside in the Adabas base source library ADA74s.MVSSRCE.

1. Execute CICEASM.

   This job preprocesses, assembles, and links this module into a staging library, and then links it with CICS entry and exit code into a CICS RPL library.

2. Use DFHCSDUP or the CEDA RDO entry panels to add the following definition to your CICS CSD file:

```
DEFINE PROGRAM(ADAENAB) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s PLTPI ENABLE ADATRUE PROGRAM)
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(CICS) EXECUTIONSET(FULLAPI)
```

   —where “s” is the system maintenance level of Adabas.

3. After defining LNKENAB to CICS, add the following entry to your PLTPI table, DFHPLT:

   DFHPLT TYPE=ENTRY,PROGRAM=ADAENAB

   This entry should follow the first DFHPLT TYPE=DELIM statement to ensure that LNKENAB will be executed in either stage II or stage III of the CICS PLTPI process. This is necessary because the CICS EXEC interface environment must be present to support the writing of console messages using the EXEC CICS WRITE OPERATOR command employed by the LNKENAB module.

4. Code an appropriate PLTPI=xx parameter in the CICS start-up data. “xx” should match the suffix value given in the DFHPLT table.

   The Adabas command-level link enhanced functions are now installed. This completes the “enhanced” installation; however, you may optionally continue with **Step 6, Installing the DISPGWA Program** on page 80.

Step 6. Installing the DISPGWA Program (Optional)

1. Execute CICDASM.

   This job preprocesses, assembles, and links this module into a staging library, then links it with CICS entry and exit code into a CICS RPL library. The final link step creates the load module ADATEST.

2. Add a CICS transaction to execute the ADATEST program. RDO may be used to do this. Sample DEFINE statements for the ADATEST (DISPGWA) program and the transaction to execute it are:

```
DEFINE PROGRAM(ADATEST) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s DISPLAY GWA PROGRAM - DISPGWA)
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE TRANSACTION(DGWA) GROUP(ADABAS)
DESCRIPTION(TRANSACTION TO DISPLAY ADABAS GWA)
       PROGRAM(ADATEST) TWASIZE(128) PROFILE(DFHCICST) STATUS(ENABLED)
       TASKDATALOC(ANY) TASKDATAKEY(CICS) STORAGECLEAR(NO)
       RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
       PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO) INDOUBT(BACKOUT)
       RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
       RESSEC(NO) CMDSEC(NO)
```

   —where “s” is the system maintenance level of Adabas.

The Adabas command-level link routine, enhanced functions, and DISPGWA program are now installed.

Installing the CICS High-Performance Stub Routine

The Adabas high-performance stub routine extends the direct call interface (DCI) facility that is available with the Adabas CICS command-level link component to applications written in languages other than Software AG's Natural (e.g., Assembler, COBOL, PL/I).

Note:
The stub routine must be used with the Adabas CICS command-level link component. The stub routine will not function properly with the Adabas CICS macro-level link component.

The DCI allows a CICS/ESA 3.2 application or above to call Adabas through the Adabas command-level link routine. The overhead incurred when the EXEC CICS LINK and EXEC CICS RETURN command set is used to transfer program control is thus avoided. Once the proper environment has been established with the initial call (IC) command from the high-performance stub or Natural 3.1 or above, the DCI permits a BALR interface to be used.

The high-performance stub routine is written in Assembler language. When linked with the application program, it serves as an interface between the application and the Adabas CICS command-level link component. The application program can then issue CALL statements to access the stub routine when executing an Adabas command.

An application at CICS/ESA 3.2 level or above derives the following advantages from the high-performance stub:

- Improved performance and throughput when issuing Adabas commands under CICS/ESA 3.2 or above due to the reduced use of CICS services related to the CICS LINK and RETURN program control mechanism.
- A call mechanism for Adabas requests under CICS/ESA 3.2 or above which is simpler than the methods normally employed to pass control with information from one program to another in the CICS environment.

Restrictions and Requirements

The following restrictions and requirements apply to the high-performance stub routine:

- **CICS/ESA 3.2 or above Required**

  The Adabas high-performance stub routine is supported under CICS/ESA 3.2 or above. Earlier versions of CICS are not supported.

  A CICS transaction work area (TWA) of at least 24 bytes must be provided to the application for the proper execution of the high-performance stub routine.

- **CICS Command-Level Link Required**

  The application program must be written using the CICS command-level interface and instructions, and may not issue any CICS macro level commands.

- **Supported Programming Languages**

  The application program may be written in ALC (Assembler language), COBOL, COBOL II, PL/I, or C. Installation verification programs (IVPs) are provided in ALC and COBOL on the distribution tape.

  Additional requirements for specific programming languages are discussed later in this chapter in the sections relating to each language.

Stub Components

Type	Member	Description
Source	ALCSIVP	source for the ALC installation verification program
	COBSIVP	source for the COBOL installation verification program
	LNCSTUB	source for the high-performance stub
Job control	JCLALCI	sample JCL to preprocess, assemble, and link the ALC installation verification program
	JCLCOBI	sample JCL to preprocess, compile, and link the COBOL installation verification program
	JCLLNCS	sample JCL to preprocess, assemble, and link the LNCSTUB (high-performance stub) routine

Installation Overview

Use the following procedure to install the Adabas CICS high-performance stub routine:

1. Edit, preprocess, assemble and link the LNCSTUB module.
2. (Optional) Modify, preprocess, compile or assemble, link, and execute the desired installation verification program (IVP).
3. Modify, preprocess, compile or assemble, link, and execute the application programs.

Step 1. Install the LNCSTUB Module

The Adabas CICS high-performance stub routine is an Assembler language module provided in source form on the distribution tape in member LNCSTUB.

Step 1 has the following substeps:

1. Supply the necessary parameters in the ADAGSET macro.
2. Change the LNCNAME field value, if necessary**.**
3. Modify member JCLLNCS.
4. Preprocess, assemble, and link the LNCSTUB module.
5. Place the LNCSTUB load module in a library that is available to your application programs when they are linked.

1.1 Code the ADAGSET Macro

Note:
For information about coding the ADAGSET macro, refer to the section ***Modifying Source Member Defaults (ADAGSET Macro)*** *starting on page 68.*

Edit the ADAGSET macro in a library that will be available in the SYSLIB concatenation when LNCSTUB is assembled.

The values given for the ADAGSET parameters are primarily for documentation purposes within the LNCSTUB module, but may be used at a later time in the stub routine at the discretion of Software AG.

1.2 Change the LNCNAME Field Value

If your Adabas CICS command-level link component program has been linked with a name other than “ADABAS”, change the constant value in the field LNCNAME to match the name used (see the ENTPT ADAGSET option description on page 69). The value in this field is used in the priming EXEC CICS LINK command issued by LNCSTUB.

1.3 Modify Member JCLLNCS

Member JCLLNCS is used to preprocess, assemble, and link the LNCSTUB module. To modify this JCL to meet your site requirements, change the JOB card in the member and the symbolic values as indicated in the following table:

Value	Description
&SUFFIX	Suffix value used for the CICS translator. The default value is “1$”.
&ASMBLR	Assembler program used to assemble the LNCSTUB source.
&M	Member name to be processed; code LNCSTUB or ALCSIVP.
&STUBLIB	A load library to contain the LNCSTUB load module. This library should be available to application programs when they are linked.
&INDEX	High-level qualifier for the CICS macro library used in the SYSLIB DD statement for the assembler.
&INDEX2	High-level qualifier for the CICS load library to use for the translator STEPLIB DD statement, and for the SYSLIB in the link step.
&ADACOML	Adabas command-level source library containing the ADACB, ADAGDEF, ADAGSET, and LNCDS copy code and macros.
&ADASRCE	Adabas source library used for additional copy code or macro expansion.
&STBSRCE	Source library containing the distributed Adabas CICS high-performance stub LNCSTUB.
&MAC1	Primary system macro library, usually SYS1.MACLIB.
&OUTC	Output class for messages, SYSPRINT, SYSOUT etc.
&REG	Step region size.
&NCAL	Value for the linkage editor NCAL parameter. The recommended value is “NCAL”.
&LSIZE	Primary and secondary table sizes used by the linkage editor.
&WORK	DASD device type to use for temporary and utility datasets.

1.4 Preprocess, Assemble, and Link the LNCSTUB Module

Because of the possible use of the 31-bit instructions, Assembler H (IEV90) or the High-Level Assembler (ASMA90) should be used to assemble the LNCSTUB module after CICS preprocessing.

In addition to the CICS macro library, the Adabas CICS command-level source library and standard Adabas source library must be provided to the SYSLIB DD statement in the assembly step.

- Do not concatenate any CICS load libraries in the SYSLIB DD statement when linking the LNCSTUB load module.
- In the SYSLIN data stream after the LNCSTUB object deck, use just the control statement

  NAME LNCSTUB(R)

- Do not include the CICS stub modules DFHEAI0 & DFHEAI1 with the LNCSTUB load module. As a result, however, the following occurs:
  - The linkage editor issues IEW462 or similar messages indicating that DFHEAI1 is an unresolved external reference;
  - The LNCSTUB module may be marked NOT EXECUTABLE by the linkage editor;
  - A condition code of 8 may be set in the link step.

  When the application program is linked with LNCSTUB, all the external references are resolved.

1.5 Make the LNCSTUB Available to Application Programs

The LNCSTUB module has an entry name of “ADABAS”, which can be used by the application program as the object of a CALL statement to pass control to LNCSTUB with a list of parameters. The language-specific calling conventions for LNCSTUB are discussed later in this chapter.

The LNCSTUB load module must be available to the link step of the application program that is to use the DCI facility.

Note:
In the same step, the CICS load library should be available; otherwise, the external references to the CICS stub modules will not be resolved.

Place the LNCSTUB load module in a library available to your application language assembler or compiler so that it will be included when the application programs are linked.

Step 2. (Optional) Install and Execute an IVP

Two installation verification programs (IVPs) are provided in source form: one for Assembler language, and one for COBOL/VS. These programs are samples for implementing the Adabas high-performance stub routine in your applications. They also provide a way of verifying the proper installation of the LNCSTUB module.

This step is optional. You may modify, preprocess, compile or assemble, link, and execute a selected installation verification program (IVP).

Step 2 has the following substeps:

1. Modify the Assembler (ALCSIVP) or COBOL (COBSIVP) source decks to provide the proper Adabas database ID and file number on your site's database for the Software AG-provided PERSONNEL file.
2. Modify the JCL provided to preprocess and compile (assemble) the desired IVP.
3. Preprocess, compile or assemble, and link the IVP using the sample JCL provided as a guide.
4. Add RDO entries to your CICS system to execute the IVPs.
5. Execute the IVPs to verify the LNCSTUB module (ALCSIVP and COBSIVP).

Field Definition Table (FDT) for PERSONNEL File

The two programs use the Software AG-provided Adabas file PERSONNEL with the following field definition table (FDT):

Level	Name	Length	Format	Options	Parent of ...
1	AA	8	A	DE, UQ	
1	AB				
2	AC	20	A	NU	
2	AE	20	A	DE	PHONDE, SUPERDE
2	AD	20	A	NU	
1	AF	1	A	FI	
1	AG	1	A	FI	
1	AH	6	U	DE	
1	A1				
2	AI	20	A	MU, NU	

Level	Name	Length	Format	Options	Parent of ...
2	AJ	20	A	NU, DE	
2	AK	10	A	NU	
2	AL	3	A	NU	
1	A2				
2	AN	6	A	NU	
2	AM	15	A	NU	
1	AO	6	A	DE	SUBDE, SUPERDE
1	AP	25	A	NU, DE	
1	AQ			PE	
2	AR	3	A	NU	SUPERDE
2	AS	5	P	NU	SUPERDE
2	AT	5	P	MU, NU	
1	A3				
2	AU	2	U	SUPERDE	
2	AV	2	U	NU	SUPERDE
1	AW			PE	
2	AX	6	U	NU	
2	AY	6	U	NU	
1	AZ	3	A	MU, NU, DE	
1	ZZ	5	U		

Note:
The two installation verification programs ALCSIVP and COBSIVP only use fields “AA” and “AE” from this FDT.

Install and Execute the Assembler IVP: ALCSIVP

The source member ALCSIVP is provided to demonstrate and verify the use of the Adabas DCI using the LNCSTUB module. This program issues a series of Adabas commands using the conventional CICS LINK/RETURN mechanism, produces a partial screen of output data, then reexecutes the same call sequence using the Adabas DCI and the LNCSTUB subprogram.

1. **Modify Source Member ALCSIVP**

   ☐ Edit the database ID and file number fields DBID (line 321) and DBFNR (line 322) to be sure they match the values needed to access the PERSONNEL file on the database you intend to use.

   ☐ Check the fields FBUFF, SBUFF and VBUFF for values consistent with your PERSONNEL file's FDT and data content.

   ☐ Check the name used in the EXEC CICS LINK statement (line 242) to be sure it matches the name of your Adabas CICS command-level link component program.

2. **Modify the JCLALCI Sample Job Stream**

   Member JCLALCI is used to preprocess, assemble, and link the installation verification program ALCSIVP. Place the load module in your CICS RPLLIB.

   To modify this JCL to meet your site requirements, change the JOB card in the member and the symbolic values as indicated in the table used in step 1 (see the section **Modify Member JCLLNCS** on page 84).

   The JCLALCI member uses one additional symbolic parameter: &CICSLIB. This is the name of your CICS RPL library.

3. **Preprocess, Assemble, and Link ALCSIVP**

   Using the modified sample JCLALCI member, preprocess, assemble, and link ALCSIVP.

4. **Add RDO Entries**

   Add the following RDO entries to your CICS system, or use the RDO facility to add the STB1 transaction to run the ALCSIVP program:

```
DEFINE PROGRAM(ALCSIVP) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s ASSEMBLER IVP FOR HIGH-PERFORMANCE STUB)
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(USER) EXECUTIONSET(FULLAPI)

DEFINE TRANSACTION(STB1) GROUP(ADABAS)
DESCRIPTION(TRANSACTION TO EXECUTE THE ASSEMBLER IVP FOR HIGH-PERFORMANCE STUB)
       PROGRAM(ALCSIVP) TWASIZE(32) PROFILE(DFHCICST) STATUS(ENABLED)
       TASKDATALOC(ANY) TASKDATAKEY(USER) STORAGECLEAR(NO)
       RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
       PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO) INDOUBT(BACKOUT)
       RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
       RESSEC(NO) CMDSEC(NO)
```

5. **Execute ALCSIVP**

Run the STB1 transaction to execute ALCSIVP. Executing ALCSIVP verifies the LNCSTUB module.

Install and Execute the COBOL IVP: COBSIVP

Member COBSIVP illustrates the use of the Adabas DCI with a COBOL program. COBIVP produces a screen showing output lines produced by a series of Adabas calls executed by the CICS LINK/RETURN facility, followed by the reexecution of these Adabas commands using the DCI.

1. **Modify Source Member COBSIVP**

- ☐ Edit the fields WORK–DBID and WORK–FNR to place the desired database ID and file number in the VALUE clauses to access the PERSONNEL file on your site's database.
- ☐ Ensure that the value in the field LINK–NAME matches the name used in your Adabas CICS command-level link component program.
- ☐ Ensure that the values (literals in the PROCEDURE DIVISION) in the following fields are consistent with the requirements of the PERSONNEL file FDT and data content you are using:
  ADABAS–FORMAT–BUFFER,
  ADABAS–SEARCH–BUFFER, and
  ADABAS–VALUE–BUFFER

2. **Modify the JCLCOBI Sample Job Stream**

Member JCLCOBI is used to preprocess, compile, and link the COBSIVP installation verification program. To modify the JCLCOBI example to meet site requirements, change the JOB card in the member and provide values for the symbolic procedure variables as described in the following table:

Value	Description
&ADALIB	Adabas load library used to provide the ADASTWA load module for the linkage editor.
&MEM	Member name to be processed; in this case, "COBSIVP".
&CICSLIB	CICS RPL library where the COBSIVP load module is placed for execution under CICS.
&COBLIB	COBOL compiler STEPLIB.

Value	Description
&INDEX	High-level qualifier for the CICS macro library used in the SYSLIB DD statement for the compiler.
&INDEX2	High-level qualifier for the CICS load library to use for the translator STEPLIB DD statement, and for the SYSLIB in the link step.
&LINKLIB	COBOL LINKLIB.
&STBSRCE	Source library containing the distributed Adabas CICS high-performance stub LNCSTUB.
&STUBLIB	A load library to contain the LNCSTUB load module. This library should be available to your application programs when they are linked.
&SYSMSG	Output class for translator messages.
&SYSOUT	Output class for SYSOUT and SYSPRINT messages.
&WORK	DASD device type to use for temporary and utility datasets.

3. **Preprocess, Compile, and Link COBSIVP**

Use the modified JCLCOBI job to preprocess, compile, and link the COBSIVP program.

☐ Assemble ADASTWA into a library available to COBOL programs when they are linked. Include the ADASTWA load module in the link of COBSIVP.

COBSIVP uses the ADASTWA subroutine when it issues Adabas calls through the standard CICS LINK/RETURN mechanism. ADASTWA is supplied in source form in the Adabas source library.

The LNCSTUB subroutine does not use ADASTWA because it places the passed Adabas parameters in the TWA. Thus, the ADASTWA routine is not required when linking COBOL applications that utilize the Adabas DCI through the LNCSTUB module.

Even though the LNCSTUB routine does not require the ADASTWA subroutine, it is included in the COBSIVP program to illustrate the “usual” way in which a COBOL application places the Adabas call parameters into the CICS TWA.

☐ Link the COBSIVP program with the LNCSTUB load module and the ADASTWA load module. Make the LNCSTUB load module available to the linkage editor to be included with the COBSIVP load module.

Note:
The CICS stub modules are also resolved in the link step.

4. **Add RDO Entries**

   Add the following RDO entries to your CICS system, or use the RDO facility to add the STB2 transaction to run the COBSIVP program:

```
DEFINE PROGRAM(COBSIVP) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s COBOL IVP FOR HIGH-PERFORMANCE STUB)
       LANGUAGE(COBOL) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(USER) EXECUTIONSET(FULLAPI)

DEFINE TRANSACTION(STB2) GROUP(ADABAS)
DESCRIPTION(TRANSACTION TO EXECUTE THE COBOL IVP FOR HIGH-PERFORMANCE STUB)
       PROGRAM(COBSIVP) TWASIZE(32) PROFILE(DFHCICST) STATUS(ENABLED)
       TASKDATALOC(ANY) TASKDATAKEY(USER) STORAGECLEAR(NO)
       RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
       PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO) INDOUBT(BACKOUT)
       RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
       RESSEC(NO) CMDSEC(NO)
```

5. **Execute COBSIVP**

   Run the STB2 transaction to execute COBSIVP. Executing COBSIVP verifies the LNCSTUB module.

Step 3. Link and Execute the Application Programs

Once the IVP programs have been successfully executed, the Adabas DCI is ready to be used with real application programs. In step 3, the application program interface (API) is coded to utilize the LNCSTUB subprogram.

Step 3 has the following substeps:

1. Modify the application programs that will utilize the Adabas CICS high-performance stub routine in accordance with the guidelines described in the following section.
2. Preprocess, compile or assemble, and link the application programs to include the LNCSTUB module.
3. Execute the application programs using the Adabas CICS high-performance stub.

Guidelines for Modifying the Application Program

The LNCSTUB load module must be linked with your application program. The application program invokes the DCI interface using a standard “batch-like” call mechanism. The LNCSTUB module makes any additional CICS requests required to pass data to the Adabas CICS command-level link component.

- **Programming Languages Supported by LNCSTUB**

  The LNCSTUB program functions with application programs written in Assembler language, VS/COBOL, COBOL II, PL/I, and C.

- **Transaction Work Area Required**

  A transaction that uses the Adabas DCI or the Adabas CICS command-level link component must provide a transaction work area (TWA) at least 28 bytes long. Failure to provide an adequate TWA will result in an ABEND U636 (abnormal termination of the task).

- **Reentrant Requirement**

  The application program may or may not be reentrant. The LNCSTUB module has been written to be reentrant, but using linkage editor parameters to “mark” the LNCSTUB load module as reentrant is not recommended.

- **CICS Requests Issued by LNCSTUB**

  The LNCSTUB module issues the following command-level CICS requests whenever it is invoked:

  EXEC CICS ADDRESS TWA
  EXEC CICS ASSIGN TWALENG
  EXEC CICS ADDRESS EIB

- **DCI Entry Point Address**

  An EXEC CICS LINK command is issued by LNCSTUB at least once to acquire the DCI entry point from the Adabas CICS command-level link component program. This address is then used for BALR access on all subsequent Adabas calls for a transaction. Thus, the calling application program must provide a fullword (4-byte) field to hold the DCI entry point address obtained by LNCSTUB. This 4-byte field is the first parameter passed to the LNCSTUB module by the call mechanism. The remaining parameters comprise the standard Adabas parameter list needed to execute an Adabas request.

- **DCI Parameter List**

  The Adabas DCI parameter list expected by the LNCSTUB program is composed of a pointer to the DCI entry point in the Adabas CICS command-level link component followed by the six pointers to the Adabas control block and buffers: format, record, search, value, and ISN.

  For information on coding the standard Adabas control block and buffers, refer to the *Adabas Command Reference Manual*.

  The parameter list offsets are summarized in the table below:

Offset	Pointer to the ...
0	DCI entry point in the Adabas command-level link component
4	Adabas control block
8	Adabas format buffer
12	Adabas record buffer
16	Adabas search buffer
20	Adabas value buffer
24	Adabas ISN buffer

  All of the parameters except the first (the DCI entry point) are built and maintained by the application program in accordance with the requirements of an Adabas call.

  The DCI entry point parameter should be set to binary zeros at the beginning of a task, and should not be modified by the application program thereafter. Software AG strongly recommends that the fields comprising the parameter list be placed in CICS storage (WORKING–STORAGE for COBOL and the DFHEISTG user storage area for Assembler) to maintain pseudo-reentrability.

Sample Parameter List for Assembler Language Program:

```
DFHEISTG    DSECT
            .
PARMLIST    DS   0F
            DS   A(DCIPTR)
            DS   A(ADACB)
            DS   A(ADAFB)
            DS   A(ADARB)
            DS   A(ADASB)
            DS   A(ADAVB)
            DS   A(ADAIB)
            .
DCIPTR      DS   F
ADACB       DS   CL80
ADAFB       DS   CL50
ADARB       DS   CL250
ADASB       DS   CL50
ADAVB       DS   CL50
ADAIB       DS   CL200
            .
DFHEIENT CODEREG=(R12),EIBREG=(R10),DATAREG=(R13)
            .
            LA   R1,PARMLIST
            L    R15,=V(ADABAS)
            BALR R14,R15
            .
            END
```

Note that the DFHEIENT macro in the Assembler example uses a DATAREG parameter of register 13. This is a strict requirement of the LNCSTUB program. When the LNCSTUB program is invoked, register 13 should point to the standard CICS save area (DFHEISA) and register 1 should point to the parameter list. The best way to ensure this standard is to code the Assembler application with a DFHEIENT macro like the one in the example.

Sample Parameter List for COBOL Language Program:

```
WORKING–STORAGE SECTION.
        .
01   STUB–DCI–PTR     PIC S9(8) COMP VALUE ZERO.
01   ADACB            PIC X(80).
01   ADAFB            PIC X(50).
01   ADARB            PIC X(250).
01   ADASB            PIC X(50).
01   ADAVB            PIC X(50).
01   ADAIB            PIC X(200).
        .
PROCEDURE DIVISION.
        .
        CALL 'ADABAS' USING STUB–DCI–PTR,
                            ADACB,
                            ADAFB,
                            ADARB,
                            ADASB,
                            ADAVB,
                            ADAIB.
        .
        EXEC CICS RETURN END–EXEC.
        .
        GOBACK.
```

- **Restrictions on Application Program Coding**

  In all other respects, the application program should be coded like a standard CICS command-level routine. As long as the DCI parameter list is correct when LNCSTUB is called, there are no restrictions on the CICS commands that an application can issue.

- **Standard Batch Call Mechanism Used**

  As shown in the Assembler and COBOL language program parameter list examples on pages 94 and 95, the call to “ADABAS” (the LNCSTUB entry point) is accomplished like a batch application. Likewise, calls for the other supported languages should be coded with their standard batch call mechanisms.

Link the Application Programs to Include the LNCSTUB Module

To properly link the LNCSTUB module with application programs, link the application program to include the LNCSTUB module and the CICS stub modules. The method for doing this varies with the programming language used for the application:

- Assembler language programs should include the DFHEAI and DFHEAI0 CICS modules;
- COBOL applications should include DFHECI and DFHEAI0.

To avoid a double reference to the DFHEAI0 module, code the linkage editor REPLACE DFHEAI0 control statement at the beginning of the SYSLIN data deck.

Linking Assembler Language Programs

For an Assembler program, the SYSLIN data looks like

```
INCLUDE DFHEAI
```

The Assembler object deck looks like

```
REPLACE DFHEAI0
INCLUDE SYSLIB(LNCSTUB)
INCLUDE SYSLIB(DFHEAI0)
NAME ALCSIVP(R)
```

When examining the cross-reference from the linkage editor, the symbol “ADABAS” must have the same starting location as the LNCSTUB module in the “link map”.

Linking COBOL Language Programs

For a COBOL program, the SYSLIN deck looks like

```
REPLACE DFHEAI0
INCLUDE SYSLIB(DFHECI)
```

The COBOL object deck looks like

```
INCLUDE SYSLIB(LNCSTUB)
INCLUDE SYSLIB(DFHEAI0)
NAME COBSIVP(R)
```

When examining the cross-reference from the linkage editor, the symbol “ADABAS” must have the same starting location as the LNCSTUB module in the “link map”.

Linking PL/I and C Language Programs

Refer to the IBM manual *CICS/ESA System Definition Guide* for information about linking PL/I and C applications under CICS.

Performance Using LNCSTUB

To obtain the best performance from applications using the Adabas direct call interface (DCI), examine how the DCI interface functions at the logical level.

A CICS application using the standard LINK/RETURN mechanism to access the Adabas link routines invokes the CICS program control service for every Adabas request made to the link routine. The LNCSTUB module permits a BALR interface to be used. A BALR interface can substantially reduce the CICS overhead required to pass control from the application program to the Adabas CICS command-level link component.

The LNCSTUB module accomplishes this by using the standard EXEC CICS LINK/RETURN mechanism to make an Initial Call (IC) to the Adabas CICS command-level link routine. The link routine recognizes this call, and returns the entry point address of the DCI subroutine to LNCSTUB. LNCSTUB must then save this address in a location that can be assured of existence throughout the duration of the invoking task. This is why the calling program must provide the 4-byte field to hold the DCI entry point address. After the DCI address has been obtained, and for as long as LNCSTUB receives this address as the first parameter passed to it on subsequent Adabas calls, LNCSTUB utilizes the BALR interface to pass control to the Adabas CICS command-level link component program.

As a consequence of this logic, the more Adabas requests made between ICs, the more efficient the application in terms of passing data to and from Adabas under CICS. In fact, pseudo-conversational applications that issue one Adabas call each time a task is invoked should not be coded to use the DCI because there will be an IC request for each Adabas command issued by the calling program.

An additional performance improvement can be realized by taking advantage of the fact that the Adabas CICS command-level link component program must be defined as resident in CICS. This fact should allow the DCI entry point to be stored across CICS tasks, making it possible for different programs to call the LNCSTUB module with a valid DCI entry point. The IC at each program startup is thus avoided. When this procedure is used, however, any change to the CICS environment that invalidates the entry point address (such as a NEWCOPY) will lead to unpredictable and possibly disastrous results.

It is imperative that at least one IC be made to the Adabas CICS command-level link component program using CICS services. This call is used to trigger the acquisition of shared storage for the Adabas user block (UB) and (in the case of migration aids) an array of register save areas. If no IC request is made, Adabas calls will not execute due to a lack of working storage, and to the fact that critical control blocks used by the link routines and the Adabas SVC are not built.

Installing Adabas with Com-plete

Certain Adabas parameters are required by Com-plete, Software AG's TP monitor, when installing Adabas. For more information, see the *Com-plete System Programmer's Manual*.

The link routine for Com-plete initialization (module/phase ADALCO) is provided in the Adabas distribution library for the OS/390 or z/OS environments. ADALCO is loaded during Com-plete initialization to service Adabas calls. The Adabas library containing ADALCO should be placed in either the OS/390 or z/OS COMPINIT library concatenation or in the VSE/ESA LIBDEF search chain (SMA-jobnumber I070).

The Adabas version 7.4 ADALCO is UES-enabled as distributed. See the section **UES-Enabled Link Routines** on page 60 for more information.

Installing Adabas with IMS

IMS/ESA Link Routines

The Adabas link routines for IMS/ESA are

- ADALNI, and
- ADALNK for batch.

ADALNI and ADALNK use the CSECT name and ENTRY directive "ADABAS" by default.

The Adabas version 7.4 ADALNI and ADALNK are UES-enabled as distributed. See the section **UES-Enabled Link Routines** on page 60 for more information.

Support for Terminals Running under a Session Manager

ADALNI and ADALNK provide the following support for IMS/DC terminals running under a **session manager**, which is defined as an environment where one "physical" user may log on to more than one "logical" terminal and execute IMS/DC messages:

- the LTERM is supported in the last eight bytes of the Adabas communication ID; and
- the SAF ID is supported for use by Adabas SAF Security if an external security package such as IBM's RACF or CA's ACF/2 is present and the user performs an IMS /SIGN ON command to log on to an IMS terminal.

In some cases it may be necessary to code a value of “TPX” for the assembler SETC symbol &IMSLVL when running under a session manager. Do not code the value “TPX” for &IMSLVL if BMP or BTS processing is to be performed by IMS messages that invoke the ADALNI routine.

Obtaining the Adabas User ID

The Adabas user ID is obtained at execution time by the ADALNI or ADALNK load module from the first eight (8) bytes of the IOPCB. The user ID is stored in the Adabas user block field UBUID.

Recommendations for IMS/ESA Users

Examine your IMS/ESA installation parameters to ensure that you are either

- running an external security package such as RACF or ACF/2; or
- using the IMS SGN=Y parameter.

If an external security package is **not** used, define at least one terminal with a required sign-on characteristic, and specify SGN=Y as part of your IMS installation and run-time parameters.

The defined terminal need not be actively used: it is only required by IMS initialization when the SGN=Y parameter is specified.

In an unsecured environment when there is no sign-on data for the IMS user, and there is not at least one terminal defined with a required sign-on (SGN=Y), the LTERM for the message being processed may contain blanks or binary zeros. The ADALNI module then terminates the IMS message (transaction) with ABEND U656 so that the Adabas user is protected from the consequences of constructing a nonunique user ID.

If an external security package **is** used, either enforce sign-on requirements or define and run the IMS/ESA system with the SGN=Y parameter.

Detailed information about the SGN=Y parameter and about installing terminals with various characteristics is provided in the appropriate IBM IMS/ESA publications.

Generating a Reentrant Version

ADALNI perform best in an IMS/ESA environment if the module runs in reentrant mode.

To generate and run a reentrant version of the ADALNI routine

1. Set the assembler Boolean variable &RENT to “1” before assembling the module.
2. Modify your application programs to provide a 256-byte reentrant work area as the seventh (7th) parameter passed on each call to ADALNI. Initialize this work area to binary zeros before calling the Adabas IMS link routine. Avoid modifying this work area between Adabas requests.

   The *Adabas Command Reference Manual* contains a sample COBOL skeleton example of the parameter list and CALL statement.

Installation Procedure

Overview

To install the Adabas IMS link routine ADALNI (SMA-jobnumber I055)

1. Edit the ADALNI source member to set the assembly variables and equate values (page 101);
2. Assemble the ADALNI module (page 103);
3. Link the ADALNI module into an appropriate execution library (page 103).

Step 1: Edit the ADALNI Source Member

The ADALNI source module must be edited before it can be assembled to provide the following information:

- Assembly language variables for the IMS level to be supported, whether the ADALNI module will be reentrant, and whether Adabas SAF Security (ADASAF) will be supported;
- Values for the assembler equates (EQU symbols) must be provided for the database ID (logical ID), Adabas SVC number, length of Adabas Review, and user information areas.

After modifying the assembler variables and the equate values, save your changes.

&RENT and &ADAESI

Two global Boolean assembly language variables are provided, &RENT and &ADAESI. Use an editor to find the SETB assembler directives in the ADALNI module and modify them to set &RENT and &ADAESI as required.

The variable &RENT should be set “on” (to a 1) if the ADALNI module is to be reentrant. See section **Generating a Reentrant Version of ADALNI** on page 100 for more information.

If you plan to use Adabas SAF Security (ADASAF) under IMS/ESA, you must

1. Set the variable &ADAESI to a “1” prior to assembling the ADALNI routine.
2. Link the ADASAF exit with the Adabas router (SVC).
3. Enforce all external security procedures for IMS transactions at user sign-on time.

The variable &ADAESI is set “on“ (to a “1”) if ADASAF is linked with the Adabas SVC that the ADALNI module will invoke during execution.

&IMSLVL

When assembling the Adabas IMS link routine that is distributed with the Adabas source library, find the assembler local variable &IMSLVL and modify the character value in the assembler SETC directive to match the level of your IMS system: V2, V3, V4, V5, or V6.

When running under some session managers, set the value for &IMSLVL to “TPX”. See page 98 for more information.

Modify Assembler Language Equates

Locate and modify the following assembly language EQU statement values:

LNUINFO={user-area | 0 }

Length of the user information area passed to the Adabas user exit B (UEXITB) and user exit A (UEXITA). Values from 0 to 32767 may be coded. The default value is 0.

LRVINFO={work-size | 0 }

Length of the Adabas Review work area provided to the Adabas Review exit (REVEXITB). The default value is 0, indicating that no Adabas Review support is required. See the appropriate Adabas Review documentation for a recommended LRVINFO value.

LOGID={dbid | 1 }

The default database ID. Values from 1 to 65535 may be provided. The default value is 1, which will be used if no value is provided in

- the Adabas control block on each call, or
- the ADARUN DDCARD input data.

SVCNR={svc-number | 249 }

The Adabas SVC number. A value from 200 to 255 may be provided. The default value is 249. The value must match the number of the Adabas SVC installed in your OS/390 or z/OS image.

Step 2: Assemble the Edited ADALNI Module

Use the IBM high-level Assembler to assemble the ADALNI routine.

The order of dataset concatenation on the assembly JCL SYSLIB DD statement is critically important:

1.	AIIvrs.MVSSRCE	the Adabas for IMS source library.
2.	ADAvrs.MVSSRCE	the standard or “base” Adabas source library.
3.	IMS.MACLIB	the IMS/ESA user MACRO interface library containing the PXPARMS macro.
4.	SYS1.MACLIB	
5.	SYS1.AMODGEN	or SYS1.MODGEN in OS/390 or z/OS systems.

Software AG recommends that you include the library IMS.MACLIB so that the IMS equates are available for the assembler.

If &IMSLVL is set to V2, it is not necessary to include the IMS macro library in the SYSLIB concatenation; however, the MVS system macro library should be included.

Step 3: Link ADALNI for IMS

Link the ADALNI module into a library that is available in your run-time concatenation.

Specifying AMODE and RMODE

Software AG also recommends that you link the Adabas IMS link routine with AMODE of 31 and a RMODE of ANY under IMS/ESA.

The addressing mode and run mode of the Adabas link routine for IMS must be chosen to match the mode of the application programs that invoke the link routine. Since the ADALNI module will not validate the parameter addresses passed to it, it is the responsibility of the user to ensure the proper addressing (AMODE) mode.

Note:
All IMS programs that are to access Adabas should be relinked to include the version 7.4 ADALNI module if IMS/ESA version 3.1 or above is used at your site.

Installing Adabas with Shadow

Shadow can be used in OS/390 or z/OS and in VSE/ESA environments. The Adabas link routine specific to Shadow, ADALNS, is provided in source form along with the job SHADASM to assemble it. SHADASM must be customized to select the Shadow macro (source) library containing the macros SAVED, $WAIT, RELOCD, RETURND, TCBD.

Selecting Options for ADALNS

Customizing the source member ADALNS means selecting the following options:

Option	Default	Specify . . .
SVCNR	0	the Adabas SVC number.
LOGID	1	the default logical database ID ranging 1-255.
NUBS	50	the number of UBs (user blocks) to be created by ADALNS. This must be high enough to handle the maximum possible number of concurrent Adabas requests.
PLINTWA	Y (yes)	“N” if the Adabas parameter list is passed in register 1 instead of at offset “0” in the Shadow TWA.
LNUINFO	0	the length for the user data to be passed from the ADALNS link routine to the Adabas user exit 4.

Shadow Table Entry for ADALNS

The user must specify the following entry in the Shadow PCT table:

```
PCT PROG=ADABAS,DISP=INITL,LANG=BAL,SAVE=YES,PURGE=YES
```

It is important that Adabas be made resident. Under Shadow, the ADABAS parameter is normally passed in the first 24 bytes of the TWA.

The user exit called from ADALNS gains control before the Adabas call (UEXITB), and can be used to modify the eight-byte UBUID field. This allows users who process the command log to have a unique terminal, since the command log presently contains only a four-byte field. This field does not contain a unique ID. The user exit could then be used to make the first four bytes unique. The user exit must create a unique user exit for each user. For Shadow, the UBUID field normally contains the constant “SHAD” in the high-order four bytes, followed by the value from ITRMTYPE.

Installing Adabas with Batch / TSO

When installing Adabas on TSO systems, the standard Adabas batch link routine (ADALNK) provides Adabas/TSO communication (SMA-jobnumber I055).

The Adabas version 7.4 ADALNK is UES-enabled as distributed. See the section **UES-Enabled Link Routines** on page 60 for more information.

However, it is important to note that user programs linked with ADAUSER also load ADARUN. ADARUN, in turn, loads other modules.

To start a user program linked with ADAUSER, the following modules must all be available from the defined load libraries for that specific TSO user at execution time:

ADAIOR	ADAMLF
ADAIOS	ADAPRF
ADALNK	ADARUN

ADALNKR : Reentrant Batch Link Routine

The ADALNKR source modules are provided in the Adabas source library to support applications where a reentrant batch link routine is desired. Several Software AG products require the use of the reentrant batch link routine and the ADALNKR load module is provided in the Adabas load library to support them.

Usage Notes

For Adabas 7.2.2 and above, the ADALNKR source module should be assembled separately to obtain the reentrant version of the batch/TSO link routine.

The ADALNKR routine differs somewhat from the nonreentrant ADALNK routine provided in the Adabas source library. It is no longer sufficient to set the &RENT Boolean variable to ‘1’ in the ADALNK module to obtain a reentrant version which is compatible with Adabas 7.2.2 and above.

Software AG still recommends that batch application programs be linked with the ADAUSER module, not ADALNK or ADALNKR. The ADAUSER load module is not reentrant, but the ADALNKR module may be linked with it as long as the application program conforms to the calling requirements described below.

5

CONNECTING UES-ENABLED DATABASES

Overview

Prior to Adabas version 7, Entire Net-Work converted all data for mainframe Adabas when necessary from ASCII to EBCDIC. Starting with version 7, Adabas is delivered with its own data conversion capability called universal encoding support (UES). Entire Net-Work detects when it is connected to a target database that converts data and passes the data through to Adabas without converting it.

For Adabas version 7.4, UES is enabled by default for the link routines ADALNK, ADALNKR, ADALCO, and ADALNI.

Note:
The use of UES-enabled link routines is transparent to applications, including applications that do not require UES translation support: it is not necessary to disable UES support. See page 60 for more information.

Load Modules

The load modules for ADALNK, ADALNKR, and ADALCO have been linked with LNKUES and the default translation tables.

LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller.

The two standard translation tables are

- ASC2EBC: ASCII to EBCDIC translation; and
- EBC2ASC: EBCDIC to ASCII translation.

The Adabas translation table pair is provided in appendix A starting on page 171.

Default or Customized Translation Tables

You may use the load modules with the default translation tables linked in, or you may prepare your own customized translation tables, reassemble the tables, and link them with the LNKUES module that is delivered.

Notes:

1. *It should only be necessary to modify these translation tables in the rare case that some country-specific character other than A–Z a-z 0–9 must be used in the additions 1 (user ID) or additions 3 field of the control block.*
2. *The load module LNKUESL delivered with earlier levels of Adabas version 7 is no longer supplied since the link jobs now specify the LNKUES module and the translation tables separately.*
3. *The LNKUES module is functionally reentrant; however, it is not linked that way in the Adabas load library.*
4. *When linking the LNKUES load module and the translation tables, the linkage editor may produce warning messages concerning the reentrant or reusability status of the linked module. These warning messages can be ignored.*

Source Modules

The ADALNK, ADALNKR, ADALCO, and ADALNI source modules have been coded to enable UES support by default when assembled:

- The &UES Boolean assembly variable is set to 1 by default.; the statement to set it to 0 has been commented out.
- The setting of the other Boolean variables and equates such as the SVC number and the database ID remain unchanged from earlier deliveries of the source modules.

Job Steps

Job library members ALNKLCO, ALNKLNK, and ALNKLNKR are set up to assemble and link the ADALCO, ADALNK, and ADALNKR modules, respectively, with the UES components in three steps:

1. Assemble the link module into the Adabas load library.
2. Assemble the two translation tables into the Adabas load library.
3. Link the link module with LNKUES and the translation tables and put the resulting load module into a “user” load library.

Calling LNKUES

LNKUES is called only on ADALNK request (X’1C’) and reply (X’20’) calls if the first byte of the communication ID contains X’01’ and the second byte does not have the EBCDIC (X’04’) bit set.

- For requests, LNKUES receives control before UEXITB.
- For replies, LNKUES receives control after UEXITA.

Required Environment

The Adabas database must be UES-enabled. See the *Adabas DBA Reference Manual* and the ADACMP and ADADEF utility chapters in the *Adabas Utilities Manual* for more information.

Connection Possibilities

UES-enabled databases are connected to machines with different architectures through Smarts, through Entire Net-Work, and optionally in an OS/390 or z/OS environment, through a direct TCP/IP link to the Adabas nucleus from web-based applications or from PC-based applications such as Software AG’s Jadabas.

Connection through Com-plete or Smarts

Because Adabas SQL Server (AQA) clients may not be strictly EBCDIC in an environment where databases are connected through Software AG's internal product Smarts (APS), the relevant Adabas link routine ADALCO is also UES-enabled by default. The sample jobstream to assemble and link the ADALCO module is ALNKLCO.

The assembled and linked ADALCO (as delivered or with customized and reassembled translation tables) is placed in the Smarts steplib or a "user" library concatenated with it.

1. Assemble the ADALCO module into the Adabas load library (SMA-jobnumber I070)

Modify the MVSJOBS member ALNKLCO to assemble and link ADALCO as follows:

- Provide all necessary jobcard information.
- Check the symbolic parameter value for version, revision level, and SM level (vrs). It must reflect the level of your Adabas source and load libraries.
- Check the dataset names for SYSLIB, SYSIN, SYSLMOD & SYSLIN in the SAGASM and LINKALL inline procedures.

```
//      JOB
//SAGASM   PROC MEM=,
//              VRS=
//ASM      EXEC PGM=ASMA90,
//     PARM='ASA,NODECK,OBJECT,USING(MAP),XREF(SHORT),TERM'
//SYSUT1   DD  SPACE=(4096,(120,120),,,ROUND),UNIT=VIO,DCB=BUFNO=1
//SYSTERM  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=ADABAS.&VRS..MVSSRCE
//         DD  DISP=SHR,DSN=SYS1.MACLIB
//         DD  DISP=SHR,DSN=SYS1.MODGEN
//SYSIN    DD  DISP=SHR,DSN=ADABAS.&VRS..MVSSRCE(&MEM)
//SYSLIN   DD  DSN=&&OBJ,SPACE=(3040,(40,40),,,ROUND),UNIT=VIO,
//             DISP=(MOD,PASS),
//             DCB=(BLKSIZE=3040,LRECL=80,RECFM=FBS,BUFNO=1)
//LINK     EXEC PGM=HEWL,REGION=2M,COND=(5,LT,ASM),
//   PARM='XREF,LIST(ALL),LET,MAP'
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//SYSLMOD  DD  DISP=SHR,DSN=ADABAS.&VRS..MVSLOAD(&MEM)
//         PEND


//*
//ADALCO   EXEC SAGASM,VRS=Vvrs,MEM=ADALCO
//LINK.SYSIN  DD  *
 MODE AMODE(31) RMODE(ANY)
 ENTRY ADABAS
 NAME ADALCO(R)
```

2. Assemble the two translation tables into the Adabas load library (SMA-jobnumber I056)

Assemble the ASCII to EBCDIC and EBCDIC to ASCII translation tables, either default or customized.

```
/*
//ASC2EBC  EXEC SAGASM,VRS=Vvrs,MEM=ASC2EBC
//LINK.SYSIN  DD  *
 MODE AMODE(31) RMODE(ANY)
 ENTRY ASC2EBC
 NAME ASC2EBC(R)
/*
//EBC2ASC  EXEC SAGASM,VRS=Vvrs,MEM=EBC2ASC
//LINK.SYSIN  DD  *
 MODE AMODE(31) RMODE(ANY)
 ENTRY EBC2ASC
 NAME EBC2ASC(R)
```

3. Link the translation tables and LNKUES into ADALCO (SMA-jobnumber I088)

Link the ADALCO, ASC2EBC, EBC2ASC and LNKUES modules into a final ADALCO module that is UES-enabled. Place this load module into a “USER.LOAD” library. Be sure to modify the &USERLIB symbol in the SYSLMOD statement to match your user load library.

```
/*
//LINKALL  PROC VRS=,USERLIB=
//LKED     EXEC PGM=HEWL,REGION=2M,COND=(5,LT),
//   PARM='XREF,LIST(ALL),LET,MAP,NCAL'
//SYSPRINT DD  SYSOUT=*
//ADALIB   DD  DISP=SHR,DSN=ADABAS.&VRS..MVSLOAD
//SYSLMOD  DD  DISP=SHR,DSN=&USERLIB
//SYSLIN   DD  DDNAME=SYSIN
//         PEND
//LINKUES  EXEC LINKALL,VRS=Vvrs,
//             USERLIB='YOUR.USER.LOADLIB'


//LKED.SYSIN  DD  *
 MODE AMODE(31) RMODE(ANY)
 INCLUDE ADALIB(ADALCO)
 INCLUDE ADALIB(LNKUES)
 INCLUDE ADALIB(ASC2EBC)
 INCLUDE ADALIB(EBC2ASC)
 ENTRY ADABAS
 NAME ADALCO(R)
/*
```

4. Make ADALCO Available to Smarts

The (re)linked ADALCO must be made available to Smarts. If you are calling Adabas version 7 and you do not have the correct LNKUES/ADALCO module, Adabas produces unexpected results: response code 022, 253, etc.

Connection through Entire Net-Work

UES-enabled databases are connected through Software AG's Entire Net-Work (WCP) using the Adabas nonreentrant batch or TSO link routine ADALNK. The sample jobstream to assemble and link the nonreentrant ADALNK module is ALNKLNK.

The assembled and linked nonreentrant, batch ADALNK (as delivered or with customized and reassembled translation tables) is placed in the Entire Net-Work steplib.

1. Assemble the ADALNK module into the Adabas load library (SMA-jobnumber I055)

Modify the MVSJOBS member ALNKLNK to assemble and link ADALNK as follows:

- Provide all necessary jobcard information.
- Check the symbolic parameter value for version, revision level, and SM level (vrs). It must reflect the level of your Adabas source and load libraries.
- Check the dataset names for SYSLIB, SYSIN, SYSLMOD, and SYSLIN in the SAGASM and LINKALL inline procedures.

```
//      JOB
//SAGASM   PROC MEM=,
//             VRS=
//ASM      EXEC PGM=ASMA90,
//     PARM='ASA,NODECK,OBJECT,USING(MAP),XREF(SHORT),TERM'
```

```
//SYSUT1   DD  SPACE=(4096,(120,120),,,ROUND),UNIT=VIO,DCB=BUFNO=1
//SYSTERM  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=ADABAS.&VRS..MVSSRCE
//         DD  DISP=SHR,DSN=SYS1.MACLIB
//         DD  DISP=SHR,DSN=SYS1.MODGEN
//SYSIN    DD  DISP=SHR,DSN=ADABAS.&VRS..MVSSRCE(&MEM)
//SYSLIN   DD  DSN=&&OBJ,SPACE=(3040,(40,40),,,ROUND),UNIT=VIO,
//             DISP=(MOD,PASS),
//             DCB=(BLKSIZE=3040,LRECL=80,RECFM=FBS,BUFNO=1)
//LINK     EXEC PGM=HEWL,REGION=2M,COND=(5,LT,ASM),
//   PARM='XREF,LIST(ALL),LET,MAP'
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//SYSLMOD  DD  DISP=SHR,DSN=ADABAS.&VRS..MVSLOAD(&MEM)
//         PEND
//*
//ADALNK   EXEC SAGASM,VRS=Vvrs,MEM=ADALNK
//LINK.SYSIN  DD  *
 MODE AMODE(31) RMODE(24)
 ENTRY ADABAS
 NAME ADALNK(R)
```

2. Assemble the two translation tables into the Adabas load library (SMA-jobnumber I056)

If you prefer to use the same translation tables that are used in Entire Net-work, change the COPY statements in ASC2EBC and EBC2ASC from UES2ASC and UES2EBC to NW2ASC and NW2EBC, respectively. After modifying the translation tables, be sure to (re)assemble them and link them with the delivered LNKUES module.

The Entire Net-Work translation table pair is also provided in appendix A starting on page 171.

Assemble the ASCII to EBCDIC and EBCDIC to ASCII translation tables, either default or customized.

```
/*
//ASC2EBC  EXEC SAGASM,VRS=Vvrs,MEM=ASC2EBC
//LINK.SYSIN  DD  *
 MODE AMODE(31) RMODE(ANY)
 ENTRY ASC2EBC
 NAME ASC2EBC(R)
```

```
/*
//EBC2ASC  EXEC SAGASM,VRS=Vvrs,MEM=EBC2ASC
//LINK.SYSIN  DD  *
 MODE AMODE(31) RMODE(ANY)
 ENTRY EBC2ASC
 NAME EBC2ASC(R)
```

3. Link the translation tables and LNKUES into ADALNK (SMA-jobnumber I088)

Link the ADALNK, ASC2EBC, EBC2ASC, LNKUES, and other user exit modules into a final ADALNK module that is UES-enabled. Place this load module into a “USER.LOAD” library. Be sure to modify the &USERLIB symbol in the SYSLMOD statement to match your user load library.

```
/*
//LINKALL  PROC VRS=,USERLIB=
//LKED     EXEC PGM=HEWL,REGION=2M,COND=(5,LT),
//   PARM='XREF,LIST(ALL),LET,MAP,NCAL'
//SYSPRINT DD  SYSOUT=*
//ADALIB   DD  DISP=SHR,DSN=ADABAS.&VRS..MVSLOAD
//SYSLMOD  DD  DISP=SHR,DSN=&USERLIB
//SYSLIN   DD  DDNAME=SYSIN
//         PEND
//LINKUES  EXEC LINKALL,VRS=Vvrs,
//             USERLIB='YOUR.USER.LOADLIB'
//LKED.SYSIN  DD  *
 MODE AMODE(31) RMODE(24)
 INCLUDE ADALIB(ADALNK)
 INCLUDE ADALIB(LNKUES)
 INCLUDE ADALIB(ASC2EBC)
 INCLUDE ADALIB(EBC2ASC)
 ENTRY ADABAS
 NAME ADALNK(R)
/*
```

4. Make ADALNK Available to Entire Net-Work

The (re)linked ADALNK must be made available to Entire Net-Work. If you are calling Adabas version 7 and you do not have the correct LNKUES/ADALNK module, Adabas produces unexpected results: response code 022, 253, etc.

Connection through a Direct TCP/IP Link

A TCP/IP link requires in addition that you link a reentrant ADALNKR module with LNKUES and your customized and reassembled translation tables and that you make the result available in the Adabas steplib.

UES-enabled databases are connected directly through TCP/IP using the Adabas reentrant batch or TSO link routine ADALNKR. The sample jobstream to assemble and link the ADALNK module is ALNKLNKR.

1. Assemble the ADALNKR module into the Adabas load library

In order to enable UES support for a database through TCP/IP, you must prepare a **modified** batch ADALNKR.

Step 1. Update the source ADALNKR:

&RENT SETB 1
SVCNR EQU nnn (hard-coded SVC number)

Step 2. Assemble and link the modified batch ADALNKR.

Modify the MVSJOBS member ALNKLNKR to assemble and link ADALNKR as follows:

- Provide all necessary jobcard information.
- Check the symbolic parameter value for version, revision level, and SM level (vrs). It must reflect the level of your Adabas source and load libraries.
- Check the dataset names for SYSLIB, SYSIN, SYSLMOD, and SYSLIN in the SAGASM and LINKALL inline procedures.

```
//      JOB
//SAGASM   PROC MEM=,
//             VRS=
//ASM      EXEC PGM=ASMA90,
//     PARM='ASA,NODECK,OBJECT,USING(MAP),XREF(SHORT),TERM'
//SYSUT1   DD  SPACE=(4096,(120,120),,,ROUND),UNIT=VIO,DCB=BUFNO=1
//SYSTERM  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=ADABAS.&VRS..MVSSRCE
//         DD  DISP=SHR,DSN=SYS1.MACLIB
//         DD  DISP=SHR,DSN=SYS1.MODGEN
//SYSIN    DD  DISP=SHR,DSN=ADABAS.&VRS..MVSSRCE(&MEM)
```

```
//SYSLIN   DD  DSN=&&OBJ,SPACE=(3040,(40,40),,,ROUND),UNIT=VIO,
//             DISP=(MOD,PASS),
//             DCB=(BLKSIZE=3040,LRECL=80,RECFM=FBS,BUFNO=1)
//LINK     EXEC PGM=HEWL,REGION=2M,COND=(5,LT,ASM),
//   PARM='XREF,LIST(ALL),LET,MAP'
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//SYSLMOD  DD  DISP=SHR,DSN=ADABAS.&VRS..MVSLOAD(&MEM)
//         PEND
//*
//ADALNKR  EXEC SAGASM,VRS=Vvrs,MEM=ADALNKR
//LINK.SYSIN  DD  *
 MODE AMODE(31) RMODE(24)
 ENTRY ADABAS
 NAME ADALNKR(R)
```

2. Assemble the two translation tables into the Adabas load library (SMA-jobnumber I056)

Assemble the ASCII to EBCDIC and EBCDIC to ASCII translation tables, either default or customized.

```
/*
//ASC2EBC  EXEC SAGASM,VRS=Vvrs,MEM=ASC2EBC
//LINK.SYSIN  DD  *
 MODE AMODE(31) RMODE(ANY)
 ENTRY ASC2EBC
 NAME ASC2EBC(R)
/*


//EBC2ASC  EXEC SAGASM,VRS=Vvrs,MEM=EBC2ASC
//LINK.SYSIN  DD  *
 MODE AMODE(31) RMODE(ANY)
 ENTRY EBC2ASC
 NAME EBC2ASC(R)
```

3. Link the translation tables and LNKUES into ADALNKR

It is now necessary to (re)link ADALNKR with LNKUES and your customized and reassembled translation tables.

Link the ADALNKR, ASC2EBC, EBC2ASC, LNKUES, and other user exit modules into a final ADALNKR module that is UES-enabled. Place this load module into a “USER.LOAD” library. Be sure to modify the &USERLIB symbol in the SYSLMOD statement to match your user load library.

```
/*
//LINKALL  PROC VRS=,USERLIB=
//LKED     EXEC PGM=HEWL,REGION=2M,COND=(5,LT),
//   PARM='XREF,LIST(ALL),LET,MAP,NCAL'
//SYSPRINT DD  SYSOUT=*
//ADALIB   DD  DISP=SHR,DSN=ADABAS.&VRS..MVSLOAD
//SYSLMOD  DD  DISP=SHR,DSN=&USERLIB
//SYSLIN   DD  DDNAME=SYSIN
//         PEND
//LINKUES  EXEC LINKALL,VRS=Vvrs,
//              USERLIB='YOUR.USER.LOADLIB'
//LKED.SYSIN  DD  *
 MODE AMODE(31) RMODE(24)
 INCLUDE ADALIB(ADALNKR)
 INCLUDE ADALIB(LNKUES)
 INCLUDE ADALIB(ASC2EBC)
 INCLUDE ADALIB(EBC2ASC)
 ENTRY ADABAS
 NAME ADALNKR(R)
/*
```

4. Make ADALNKR Available to the Adabas Nucleus

The (re)linked ADALNK must be made available to the Adabas nucleus.

If you are calling Adabas version 7 directly through a TCP/IP link and the correct ADALNKR is not available to the Adabas nucleus, Adabas produces unexpected results: response code 148, empty buffers, etc.

Activating the TCP/IP Link

To activate a direct TCP/IP link to the Adabas nucleus

☐ Set the ADARUN parameter TCPIP=YES.

☐ Specify a universal resource locator (URL).

Specifying a URL

The URL is a 20-byte address that conforms to the RFC specification for URLs.

You can specify the URL required to activate the direct TCP/IP link in the ADARUN parameter TCPURL as follows:

```
ADARUN PROG=ADANUC,TCPIP=YES,TCPURL=api-name://stackid:port-number
```

—where

api-name	is a 1–3 character value identifying the application programming interface (API) to use. The APIs for the IBM TCP/IP stack (HPS, OES) are currently supported.
stackid	is a 1–8 character value identifying the stack to use: • for the HPS API, this is the name of the TCP/IP started task. • for the OES API, no value is needed. • for the ILK API, this is the subsystem identifier.
port-number	is a 1–5 character number in decimal notation.

Examples

```
ADARUN PROG=ADANUC,TCPIP=YES,TCPURL=HPS://STACKNAME:1234

ADARUN PROG=ADANUC,TCPIP=YES,TCPURL=OES://:1234

ADARUN PROG=ADANUC,TCPIP=YES,TCPURL=ILK://ILZ5:1234
```

Managing URLs

Optionally, you can specify the first and additional URLs using the operator command TCPIP:

```
TCPIP={ OPEN=url|CLOSE=url | CLOSE }
```

—where ‘url’ is the URL for the TCP/IP link you want to open or close and has the same format as the ADARUN TCPURL parameter:

api-name://stackid:port-number

The command allows you to open or close a TCP/IP link to the Adabas nucleus or to close all links. It can only be used when ADARUN TCPIP=YES and all conditions for that setting have been met. This command can be used to close the URL set in the ADARUN TCPURL parameter, or to open/close additional TCP/IP links.

Examples

TCPIP=OPEN=ILK://ILZ5:1234
TCPIP=CLOSE=ILK://ILZ5:1234

To close all open URLs:

TCPIP=CLOSE

6

DEVICE AND FILE CONSIDERATIONS

This chapter provides information for the following device- and system file-related topics:

- defining new devices.
- using VSAM files in OS/390 or z/OS environments.

Supported Device Types

The standard characteristics of the device types supported by Adabas on OS/390 or z/OS are summarized in the following table:

		Adabas Block Sizes : RABNs per Track						
Device	**Trks/ Cyl**	**ASSO**	**DATA**	**WORK**	**PLOG/ RLOG**	**CLOG**	**TEMP/ SORT/ DSIM**	**Notes**
0512	16	2044:8	4092:4	8192:2	8192:2	8192:2	8192:2	
3310	11	2044:8	4092:4	4096:4	4096:4	4096:4	8192:2	
3330	19	1510:8	3140:4	4252:3	4252:3	3156:4	3140:4	
3340	12	1255:6	2678:3	3516:2	3516:2	3516:2	3500:2	
3350	30	1564:11	3008:6	4628:4	4628:4	3024:6	3008:6	
3370	12	2044:15	3068:10	5120:6	5120:6	3072:10	7680:4	
3375	12	2016:15	4092:8	4096:8	4096:8	4096:8	8608:4	
3380	15	2004:19	4820:9	5492:8	5492:8	4820:9	7476:6	3
3390	15	2544:18	5064:10	5724:9	5724:9	5064:10	8904:6	3
8345	15	4092:10	22780:2	22920:2	22920:2	22920:2	22920:2	
8350	30	3008:6	6232:3	9442:2	9442:2	9442:2	9442:2	1
8380	15	3476:12	6356:7	9076:5	9076:5	9076:5	9076:5	1
8381	15	3476:12	9076:5	11476:4	11476:4	9076:5	9076:5	1
8385	15	4092:10	23292:2	23468:2	23468:2	23468:2	23468:2	1

		Adabas Block Sizes : RABNs per Track						
Device	**Trks/ Cyl**	**ASSO**	**DATA**	**WORK**	**PLOG/ RLOG**	**CLOG**	**TEMP/ SORT/ DSIM**	**Notes**
8390	15	3440:14	6518:8	10706:5	10706:5	8904:6	8904:6	1
8391	15	4136:12	10796:5	13682:4	13682:4	8904:6	18452:3	1
8392	15	4092:12	12796:4	18452:3	18452:3	18452:3	18452:3	1
8393	15	4092:12	27644:2	27990:2	27990:2	27990:2	27990:2	1
9332	6	2044:10	4092:5	5120:4	5120:4	10240:2	10240:2	2
9335	6	2556:14	3580:10	5120:7	5120:7	7168:5	7168:5	
9345	15	4092:10	7164:6	11148:4	11148:4	22920:2	22920:2	3

Notes:

1. *The 8350, 838n, and 839n are pseudodevice types physically contained on a 3350, 3380, and 3390 device, respectively, but for which some or all of the standard block sizes are larger.*
2. *The number of tracks per cylinder listed here is artificial.*
3. *The IBM RAMAC 9394 emulates devices 3390 Model 3, 3380 Model K, or 9345 Model 2.*

Support for VSAM Datasets

VSAM support is available only on OS/390 and z/OS.

To support VSAM datasets, the following table shows the CISZ values for the Adabas components on the 3380/90 devices:

Device	**ASSO**	**DATA**	**WORK**	**PLOG/ RLOG**	**CLOG**	**TEMP/ SORT/ DSIM**
3380	2048	5120	5632	5632	5120	7680
3390	2560	5120	6144	6144	5120	9216

The VSAM device types 5555, 6666, 7777, 8888, and 9999 are dynamic device types that depend on the user definition.

A VSAM user can determine the RABN size currently in use from message ADAI64.

ECKD Devices

Adabas supports ECKD DASD devices such as the IBM 3390 with the 3990 controller and ESCON channels.

During an open operation, ADAIOR determines which DASD device types are being used for the ASSO, DATA, WORK, SORT, and TEMP datasets. At that time, Adabas issues an informational message for each Adabas database component, where “type” is the component:

ADA164 ... FILE DDtype **HAS BEEN OPENED IN** ckd/eckd **MODE – RABN SIZE** rabn-size

Important:
Software AG strongly recommends that you avoid mixing ECKD and CKD extents within a file, because the file will be opened only in CKD mode. Mixing extents could degrade performance when file I/O operations are performed.

Adding New Devices

Support for new device types that include user-defined block sizes can be implemented in ADAIOR by modifying one of the table of device-constant entries (TDCEs) reserved for this purpose.

A TDCE is X’40’ bytes long and the first free TDCE can be identified by X’0000’ in its first two bytes (TDCDT).

For all versions of Adabas prior to version 6.2, the address of the first TDCE is at offset ADAIOR+ X‘34’.

For Adabas version 6.2, TDCE entries are in the ADAIOR CSECT TDCON: the first TDCE entry is at offset 0; the first free TDCE entry is at offset X‘400’.

For Adabas version 7.1, TDCE entries are in the ADAIOS CSECT TDCON: the first TDCE entry is at offset 0; the first free TDCE entry is at offset X‘580’.

This information is valuable when adding an additional TDCE entry.

Information to be Zapped into the First Free ADAIOR TDCE

The information in the following tables must be zapped into the first free TDCE. The rules described in the section **General Rules for Defining Device Block Sizes** starting on page 125 must be followed when changing the TDCE.

Label	Offset	Contents
TDCDT	00	Device type in unsigned decimal (X'3385'), must be numeric, and unique among all TDCEs.
TDCKSN	02	Constant set number: must be uniquely chosen from the values X'2B' or X'2E'.
TDCF	03	The flag bit must be set—TDCFCKD (X'40') for CKD devices or TDCFECKD (X'60') for ECKD devices.
TDCDT1	04	(see note 1)
TDCDT2	05	(see note 1)
TDCDT3	06	(see note 1)
TDCDT4	07	(see note 1)
TDCMSBS	08	Refer to the section **Maximum Sequential Block Size** on page 125.
TDCTPC	0A	Number of tracks per cylinder.
TDCCIPT	0C	(see note 2)
TDCBPCI	0E	(see note 2)
TDCABPT	10	Number of Associator blocks per track.
TDCABS	12	Associator block size.
TDCACPB	14	(see note 2)
TDCDBPT	16	Number of Data Storage blocks per track.
TDCDBS	18	Data Storage block size.
TDCDCPB	1A	(see note 2)
TDCWBPT	1C	Number of Work blocks per track.
TDCWBS	1E	Work block size.
TDCWCPB	20	(see note 2)
TDCTSBPT	22	(see note 2)
TDCTSBS	24	TEMP or SORT block size.

Label	Offset	Contents
TDCTSCPB	26	(see note 2)
TDCPBPT	28	Number of PLOG blocks per track.
TDCPBS	2A	PLOG block size.
TDCPCPB	2C	(see note 2)
TDCCBPT	2E	Number of CLOG blocks per track.
TDCCBS	30	CLOG block size.
TDCCCPB	32	(see note 2)

Notes:

1. *One or more operating-system-dependent codes for identifying the device type: OS/390, z/OS, the UCB unit type from UCBTBYT4.*
2. *Not used for OS/390 or z/OS operating systems.*

General Rules for Defining Device Block Sizes

The following general rules must be followed when defining Adabas device block sizes:

- All block sizes must be multiples of 4.
- A single block cannot be split between tracks (i.e., the block size must be less than or equal to the track size.

Maximum Sequential Block Size

When adding new devices, the maximum sequential block size must also be specified. The value to be set to the maximum sequential block size is TDCMSBS, located at offset X’08’ from the beginning of the ADAIOR TDCE table.

Depending on the device type, the TDCMSBS value should be as follows:

Device Type	Maximum Block Size
0512	32760
3310	32760
3330	13030

Device Type	Maximum Block Size
3340	8368
3350 (8350)	19069
3370	32760
3375	17600
3380 (8380/81)	23476
339n	27998
8380/1/5	23476
839n	27998
9332	32760
9335	32760

Note that on some devices, it may be most efficient to use smaller block sizes (for example, to specify 23476 for the 3380, but with two blocks per track).

Rules for Associator and Data Storage Block Sizes

- Associator block size must be greater than one-fourth the size of the largest FDT, and should be large enough to accept definitions in the various administrative blocks (RABN 1 - 30) and in the FCB;
- The block sizes for Associator and Data Storage should be a multiple of 256, less four bytes (for example, 1020) to save Adabas buffer pool space.
- The Associator and Data Storage block sizes must be at least 32 less than the sequential block size.
- Data Storage block size must be greater than: (maximum compressed record length + 10 + padding bytes).

Rule for Work Dataset Block Size

The Work block size must be greater than either (maximum compressed record length + 100) or (Associator block size + 100), whichever is greater.

Rules for TEMP/SORT Dataset Block Sizes

If ADAM direct addressing is used:

size > (maximum compressed record length + ADAM record length + 24);

size > 277 (maximum descriptor length + 24)

- However, TEMP and SORT are generally read and written sequentially; therefore, the larger the TEMP/SORT block size, the better.
- Block sizes for TEMP and SORT must be greater than the block sizes for Data Storage.

Rules for PLOG or SIBA Block Sizes

Note:

The use of 3480/3490 tape cartridge compression (IDRC) is not recommended for protection log files. The ADARES BACKOUT function will run at least twice as long under OS/390 or z/OS when processing compressed data.

- PLOG or SIBA block size must be greater than or equal to the Work block size.
- It is also recommended that PLOG/SIBA be defined larger than the largest Data Storage block size. This avoids increased I/O caused by splitting Data Storage blocks during online ADASAV operations.

The block size (BLKSIZE) of a sequential file is determined as follows:

```
     if PTTF(JCL) then BLKSIZE is taken from file assignment statement or la-
bel;
     if PTTMBS > 0 then BLKSIZE = PTTMBS;
     if PTTMBS = 0 then
       if tape then BLKSIZE = 32760;
       else BLKSIZE = TDCMSBS;
     else if BLKSIZE in file assignment statement or label then use it;
          if PTTF(OUT) then
            if QBLKSIZE > 0 then BLKSIZE = QBLKSIZE;
            if tape then BLKSIZE = 32760;
            else BLKSIZE = TDCMSBS;
          else error.
```

Note:
QBLKSIZE is an ADARUN parameter.

Sequential Protection Log Block Size in I_PPT

In addition, the sequential protection log block size may have to be increased in the corresponding PTT entry in CSECT I_PTT of the load module ADAIOR.

The address of the first PTT entry is contained in the fullword at ADAIOR+X‘4C8’.

PTT entries begin at offset 0 into CSECT I_PTT.

Each PTT entry is X’10’ bytes long and has the structure given below:

Label	Offset	Contents		
PTTPN	00	Program number		
PTTFT	01	File type		
PTTN	02	DD name characters 2 - 8		
PTTF	08	Flags:		
		OUT	(X’80’)	output
		BSAM	(X’40’)	BSAM
		BACK	(X’20’)	read backwards
		JCL	(X’10’)	BLKSIZE/LRECL/RECFM taken from DATADEF statement or label
		UNDEF	(X’04’)	undefined record format
		VAR	(X’02’)	variable record format
–	09	Reserved		
PTTMBSZ	0C	Maximum block size		

The PTT entry for the sequential protection log can be identified by X‘12F1’ in its first two bytes.

Installing Adabas Using VSAM Datasets

This section presents information needed to install Adabas on OS/390 or z/OS systems using VSAM as the access method for containing Adabas data. Software AG provides a VSAM interface as an alternative to using EXCP with BDAM container files.

Suggested Additional VSAM Information Sources

Software AG recommends that you have the following reference manuals when dealing with VSAM files. Later references are made in this section to these manuals:

- *IBM Access Method Services*
- *IBM VSAM Administration Guide, Macro References*

See the list of manuals in the introduction for more specific information.

Defining VSAM Datasets

The following topics describe the VSAM file types used for Adabas files, and how to define and delete those files with the IDCAMS utility.

VSAM File Types

There are four types of VSAM container files:

KSDS	key sequential datasets
ESDS	entry sequence datasets
RRDS	relative record datasets
LDS	linear datasets

Adabas uses RRDS or the optional LDS as the VSAM container file type that must be defined.

Normally, files are seen as containing records that can be read and written by the application program. Adabas uses the VSAM record to hold a block of compressed Adabas data. This means that the VSAM file contents is identical with that of its BDAM equivalent; only the access method is different. The RRDS and LDS VSAM file types were chosen because of their similarity to the current BDAM file structure.

Defining a VSAM Dataset with the IDCAMS Utility

To define a VSAM dataset, the IBM-supplied utility IDCAMS is used. This utility and its parameters are discussed in the manual *IBM Access Method Services*. It is not the intention here to describe IDCAMS in detail, but only those aspects that relate to Adabas requirements.

The following is an example job set-up for defining a VSAM dataset for Adabas:

```
//IDC01       EXEC     PGM=IDCAMS
//SYSPRINT    DD       SYSOUT=*
//SYSOUT      DD       SYSOUT=*
//SYSIN       DD       *
  DEFINE CLUSTER ( NAME (EXAMPLE.ASSOR1) VOL (VOLXXX) -
         CYLINDER(100) NUMBERED) -
  DATA (NAME (EXAMPLE.ASSOR1.DATA) -
        SHAREOPTIONS ( 3 3 ) -
        CISZ (2048) -
        RECORDSIZE (2004 2004) )
/*
//
```

The first SYSIN statement defines the VSAM cluster. The information that follows the SYSIN statement (within the outer parentheses) describe the VSAM file. The following is a short description of the parameters:

Parameter	Description
NAME	Name assigned to the dataset and used to refer to the dataset in later jobs; for example, the following data definition refers to the name defined in the above example: `//DDASSOR1 DD DSN=EXAMPLE.ASSOR1,DISP=SHR`
VOL	Volume or volumes on which this file is contained.
CYLINDER	DASD space required for this component. This may also be specified as RECORDS, TRACKS, etc.
NUMBERED	Identifies the file type as RRDS.
DATA (NAME)	Internal name describing the Data component of the VSAM file.
SHAREOPTIONS (3 3)	Defines how this dataset is to be shared among other users. See the *IBM Access Method Services* manual for more information.
CISZ	Internal VSAM control interval size for this dataset. See the section **Defining Control Interval Sizes** on page 132.
RECORDSIZE	VSAM record size; for Adabas use, this is the Adabas block size (see appendix A, *Adabas Operations Manual*).

The “–” character indicates continuation on the next job statement line.

Deleting a VSAM Dataset

The IDCAMS utility is also used to delete a VSAM dataset. The following is an example of a job for deleting a VSAM file:

```
//IDC01       EXEC     PGM=IDCAMS
//SYSPRINT    DD       SYSOUT=*
//SYSOUT      DD       SYSOUT=*
//SYSIN       DD       *
  DELETE (EXAMPLE.ASSOR1)
/*
//
```

Multiple IDCAMS Operations

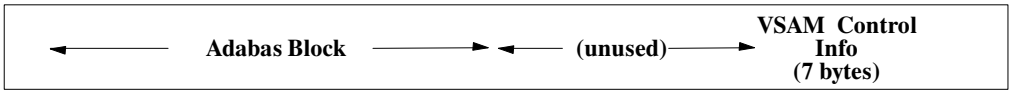
You can combine more than one IDCAMS utility operations. For example, you can delete and define a cluster or multiple clusters in one execution step. The following is an example:

```
//IDC01       EXEC     PGM=IDCAMS
//SYSPRINT    DD       SYSOUT=*
//SYSOUT      DD       SYSOUT=*
//SYSIN       DD       *
  DEFINE CLUSTER ( NAME (EXAMPLE.ASSOR1) VOL (VOLXXX) –
         CYLINDER(100) NUMBERED) –
  DATA (NAME (EXAMPLE.ASSOR1.DATA) –
        SHAREOPTIONS ( 3 3 ) –
        CISZ (2048) –
        RECORDSIZE (2004 2004) )
  DELETE (EXAMPLE.DATAR1)
  DEFINE CLUSTER ( NAME (EXAMPLE.DATAR1) VOL (VOLXXX) –
         CYLINDER(100) NUMBERED) –
  DATA (NAME (EXAMPLE.DATAR1.DATA) –
        SHAREOPTIONS ( 3 3 ) –
        CISZ (5120) –
        RECORDSIZE (4820 4820) )
/*
//
```

Defining Control Interval Sizes

A “control interval” is an area in which VSAM manages a record or group of records. VSAM maintains locks at the control-interval level; if an update is in progress for a record within a control interval, all records in that control interval are also locked, and cannot be accessed or changed. For this reason, **only one record per control interval should be defined for Adabas VSAM files**, since Adabas must be allowed to update or access any block on the database at any given time.

The following diagram shows a control interval containing an Adabas block:



How large the control interval is, depends on the record size. For records less than 8 KB, the control interval is defined in multiples of 512 bytes; for records equal to or larger than 8 KB, the control interval is defined in multiples of 2048 bytes.

To ensure that the correct CISZ is coded in your IDCAMS DEFINE request, use the following formula:

For records less than 8 KB:

$$\text{resulta} = \frac{\text{recordsize} + (\text{7-byte CI overhead})}{512}$$

Round “resulta” up to the next higher number, then calculate CISZ:

$$\text{CISZ} = \text{resulta (rounded)} \cdot 512$$

For records equal or larger than 8 KB:

$$\text{resulta} = \frac{\text{recordsize} + (\text{7-byte CI overhead})}{2048}$$

Round “resulta” up to the next higher number, then calculate CISZ:

$$\text{CISZ} = \text{resulta (rounded)} \cdot 2048$$

The following table shows the record sizes and CISZ values for the Adabas components on the 3380/90 devices:

Device	ASSO	DATA	WORK	PLOG/ RLOG	CLOG	TEMP/ SORT
3380	2004:2048	4820:5120	5492:5632	5492:5632	4820:5120	7476:7680
3390	2544:2560	5064:5120	5724:6144	5724:6144	5064:5120	8904:9216

Using Existing Adabas Device Definitions

It is possible to use existing devices and device characteristics within the Adabas VSAM interface. No DEVICE-related changes are required in the ADARUN control parameters. The only requirement is that the IDCAMS RECORDSIZE parameter be identical to the block size of your existing BDAM component. However, it should be taken into account that this may lead to wasted disk space.

For example, a 3380 definition for the Associator requires a block size of 2004 bytes, plus seven bytes overhead per control interval and only one record per control interval. The actual use of the control interval is 2011 bytes, resulting in an unused area of 37 bytes (2048 – 2011 = 37). This unused space generally requires that the VSAM physical space allocation for the dataset be larger than its BDAM equivalent.

It is not necessary that the VSAM datasets be on the same device type as that defined on the present ADARUN statements. For example, you may define the RECORDSIZE parameter for IDCAMS as a 3390 Adabas block size, while the VOL parameter points to a 3350 physical device type. In addition, VSAM allows the cluster to span different physical device types, while appearing to Adabas as a single device type.

Defining Your Own Device Characteristics

When you define your own block sizes, the Adabas VSAM interface detects this and dynamically creates a device type of 9999 for the DD/xxxxR1 Adabas component. Or, if the VSAM interface detects an ADARUN DEVICE=9999 parameter, the VSAM file information is obtained from the VSAM catalog entry.

By defining your Adabas datasets with your own RECORDSIZE definition, you can ensure the best use of VSAM's control intervals. However, you must adhere to the guidelines for Adabas block sizes as defined in the section **General Rules for Defining Device Block Sizes** starting on page 125, except for the restriction that prohibits splitting a block between tracks. This can be done with the VSAM interface.

DD/xxxxR2 Adabas components (DD/PLOGR2, DD/ASSOR2, etc.) are defined using a dynamic device type of 8888, DD/xxxxR3 uses 7777, DD/xxxxR4 uses 6666, and DD/xxxxR5 uses 5555. For utilities that require DATADEV or ASSODEV, it is important to provide the appropriate dynamic device types according to this system.

The DD/xxxxR2 – R5 datasets are only required for different Adabas block size definitions; they are no longer needed for defining VSAM files on different physical device types alone. A single component may span physically different devices in the VSAM interface, while still appearing to Adabas as being on the same device type.

Mixing VSAM and BDAM Components

VSAM and BDAM components can be mixed. For example, you can have an Associator in a VSAM file with the rest of the Adabas components in BDAM files. But you cannot mix VSAM and BDAM within the same component. If, for example, you have DD/ASSOR1 and R2 defined, they must both be either VSAM or BDAM files. Any Adabas component that currently resides on a BDAM file may be redefined on a VSAM dataset. This includes Associator, Data, Work, PLOG, CLOG and the RLOG.

Converting Adabas BDAM Components to VSAM

There are two ways to convert Adabas Associator (ASSO) and Data Storage (DATA) components to VSAM. The simplest method, described next, may require more DASD space for the VSAM components. The second method takes more time, but may save on DASD space.

Method 1

1. Run the ADAREP (database report) utility on the existing database with its BDAM components.
2. Run ADASAV SAVE on the database.
3. Run the IDCAMS utility to DEFINE the cluster or clusters for the Associator and/or Data Storage components being converted. Specify a RECORDSIZE that is consistent with these components, and a RECORDS value using the number of blocks indicated in the ADAREP output's "database physical layout" section for those components being converted. Remember to add one track's worth of blocks to the size reported there.
4. Following successful IDCAMS operation, run the ADAFRM utility on the VSAM files to format them. Remember to use the same device types as were defined on the BDAM database.
5. Now, run the ADASAV restore function on the new VSAM datasets.
6. Convert all other nucleus and utility job control statements to apply to the VSAM datasets.

Method 2

1. Run the ADAORD RESTRUCTUREDB function on the BDAM-based database. Do not specify an Associator device type with ASSODEV different from the existing Adabas block size and device type definitions.
2. Run IDCAMS to allocate the VSAM files, using your own RECORDSIZE and size definitions. Define all Adabas Data Storage and/or Associator components.
3. Run the ADAFRM utility to format the VSAM files created in step 2. When using existing Adabas block sizes, use the existing device definition; otherwise, specify DEVICE=9999 to indicate dynamic device usage (see the section **Defining Your Own Device Characteristics** on page 133 for more information).
4. Run the ADADEF DEFINE function on the VSAM files, and specify ASSOSIZE/DATASIZE according to the result of step 2, above, minus one track's worth of blocks.
5. Run the ADAORD STORE function on the VSAM files, and specify the correct device types. Use device type "9999" for dynamic device usage (see the section **Defining Your Own Device Characteristics** on page 133 for more information).
6. Change all other nucleus and utility job control to specify the VSAM datasets defined and formatted in steps 2 through 5. Remember to change any device specifications to "9999" if you are using dynamic device definition (see the section **Defining Your Own Device Characteristics** on page 133 for more information).

Converting WORK, PLOG, CLOG and RLOG Components

1. Execute IDCAMS to define the cluster, using either the existing Adabas block sizes or your own RECORDSIZE definitions.
2. Execute ADAFRM to format the VSAM datasets.
3. Use the VSAM file or files in place of their BDAM counterparts in all Adabas nucleus and utility job control statements.

VSAM File Storage Requirements

The Adabas VSAM interface makes use of BDAM user buffering and VSAM control interval processing to minimize VSAM overhead. However, this requires that the Adabas VSAM interface acquire storage to manage the contents of VSAM control intervals.

The Adabas VSAM interface uses up to 255 areas per Adabas file to manage VSAM control intervals. Each area is equivalent to the CISIZE specified in the IDCAMS definition for the file. These control interval areas are acquired dynamically while the nucleus or utility is executing; this minimizes the amount of storage required, based on the I/O response times of your environment. For example, in an environment where I/O performance is optimized, it is possible that only seven to ten of these CIAREAs would be needed to handle concurrent asynchronous VSAM requests.

In 31-bit addressing mode, these buffers are allocated above 16 MB. Software AG therefore recommends that you run with AMODE=31 when using the VSAM interface.

Allocating VSAM Datasets on Multiple Volumes

Allocating VSAM datasets across multiple volumes is done by specifying secondary space allocations as well as the volume serial numbers that will contain the VSAM dataset. The following is a sample IDCAMS execution for defining EXAMPLE.ASSOR1 on volumes VOLXXX and VOLYYY with a primary space allocation of 100 cylinders and a secondary space allocation of 50 cylinders:

```
//IDC01   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SYSIN  DD *
  DEFINE CLUSTER ( NAME (EXAMPLE.ASSOR1) VOL (VOLXXX VOLYYY) -
         CYLINDER(100 50) NUMBERED) -
  DATA (NAME (EXAMPLE.ASSOR1.DATA) -
         SHAREOPTIONS ( 3 3 ) -
         CISZ (2048) -
         RECORDSIZE (2004 2004) )
```

If the secondary allocation of 50 cylinders can be obtained on VOLXXX, it will be taken from that volume. Allocation of 50 cylinders will continue until VOLXXX can no longer satisfy an allocation. Then, space is acquired from VOLYYY.

The primary allocation must be acquired from VOLXXX; otherwise, IDCAMS will fail. The maximum number of VSAM extents is 123. On large databases, you must be careful to avoid fragmented VSAM file allocations that could cause the limit of 123 extents to be exceeded. For extremely large databases or for those requiring more than 123 extents, consider allocating additional database components (DD/xxxxR2 – R5).

VSAM Limitations

VSAM files can be defined to include up to 123 extents, or a maximum size of 4,294,967,296 bytes (4 gigabytes). Therefore, Adabas is permitted a maximum of 20 gigabytes of Associator (ASSO) and 20 gigabytes of Data Storage (DATA) components through the use of DD/xxxxR1 – DD/xxxxR5 datasets, and up to 4 gigabytes for all other Adabas components.

Comparison of EXCP and VSAM on High-Capacity Disk Drives

Newly developed DASD (disk) devices have capacities considerably larger than any previous devices. The IBM 3390 Model 9 is one example. Such devices contain more than 65,535 tracks per volume, which makes it impossible to allocate a complete volume as a single dataset, regardless of the number of extents. The 65,535 track limit is imposed by the operating system for most access methods.

To make use of a whole volume for ASSO or DATA files, it becomes necessary to allocate more than one container dataset per volume, each dataset not exceeding 65,535 tracks. These datasets would then be assigned to separate DATARn or ASSORn DD statements in Adabas JCL procedures. Since DATA and ASSO can each have up to five containers, this technique allows up to 327,675 tracks total for each. On a 3390-type device, this can be more than 18 GB.

If larger DATA or ASSO files are required, it is necessary to allocate one or more container files on multiple volumes. The operating system permits a single dataset to span up to 59 volumes, as long as the total number of tracks allocated on each volume does not exceed 65,535. This allows DATA and ASSO each to be as large as 19,332,825 tracks, which can be more than 1 TB on 3390-type devices.

When deciding between EXCP and VSAM for such a device, the factors that should be considered are

- performance;
- maximum capacity; and
- ease of maintenance.

VSAM permits easier handling and improved SMS integration, but at the cost of somewhat lower performance. Also, since there can only be one VSAM cluster per operating system file (i.e., ASSOR1, etc.), this limits the total Associator or Data Storage size to 20 gigabytes. The maximum size for the Work dataset is four gigabytes.

The capacity of high-capacity devices themselves may also restrict the choice. Unlike a BDAM file, a single VSAM cluster cannot exceed four gigabytes. This means that a single VSAM cluster cannot reference the complete volume on such a device (for example, the 3390 Model 9 has a maximum capacity of 8.5 gigabytes).

On the other side, EXCP offers high volume and performance, but with the disadvantage of requiring more maintenance.

It should also be noted that certain high-capacity drives have a slower data transfer rate. The performance impact of the data transfer rate must be taken into consideration when choosing the access method to be used on such devices.

7

INSTALLING THE AOS DEMO VERSION

This chapter tells you how to manually install the Adabas Online System (AOS) demo version on an OS/390, z/OS, or a FACOM MSP system. To install this facility on systems that use Software AG's System Maintenance Aid (SMA), refer to the chapter of this manual devoted to installing Adabas in your particular operating environment.

Notes:

1. *To install the full version selectable unit AOS, see the Adabas Online System Manual (AOSvrs–030IBB).*
2. *Demo versions of Adabas Vista (AVI), Adabas Fastpath (AFP), Adabas SAF Security (AAF), and Adabas Transaction Manager (ATM) are automatically installed when you install either the demo or full version of AOS.*

The AOS demo version requires Natural version 3.1 or above.

For information about SMA, see the *System Maintenance Aid Manual* (SMAvrs-030IBB).

Installing

To install the AOS demo version without the System Maintenance Aid

1. For a Com-plete or CICS environment, link the correct object module with the Natural TP nucleus.

   If a split Natural nucleus is to be installed, the AOSASM module **must** be linked to the shared portion of the nucleus and **not** to the thread portion.

2. Perform a Natural INPL.

   The tape containing the AOS demo version facility contains an INPL-formatted dataset in Natural 3.1. The programs for the AOS demo version are stored in library SYSAOS.

3. Load the ADA error messages using the Natural utility ERRLODUS.

   The error messages are stored in an ERRN-formatted dataset included on the tape.

   See the *Natural Utilities Manual* for information about the ERRLODUS utility.

4. Finally, execute the AOS demo version by logging on to the application library SYSAOS and entering the command “DBMENU”.

Installing with Natural Security

If Natural Security is installed, define at least the library SYSAOS to it.

Define the following libraries as needed:

- For Adabas Vista: SYSAVI and SYSMVvrs
- For Adabas Fastpath: SYSAFP and SYSMWvrs
- For Adabas SAF Security: SYSAAF and SYSMXvrs
- For Adabas Transaction Manager: SYSATM and SYSMTvrs

Software AG recommends you define SYSAOS and any other libraries you may define as protected.

Specify the start-up program for SYSAOS as DBMENU. Do not specify a start-up program name for the other libraries.

Natural Security must be installed before implementing Adabas Online System Security. See the *Adabas Security Manual* for more information. For information about installing Natural Security for use with AOS Security, see the *Natural Security Manual*.

Natural Security includes the ability to automatically close all open databases when the Natural command mode’s LOGON function of the AOS demo version is invoked.

Setting the AOS Demo Version Defaults

Parameters that control the operation of the AOS demo version can be set at installation time by changing the defaults in the Natural program ADVUEX1. The table below lists the parameters and possible values. Default values are underlined:

Parameter	Valid Values / Default	Function
AOS-END-MSG	Yes (Y) / No (N)	Display the AOS demo version end-of-session message?
AOS-LOGO	Yes (Y) / No (N)	Display the AOS demo version logo?
CPEXLIST	No (N): normal list Yes (Y): extended	Display extended checkpoint list?
MAX-AC-IOS	0–999999 (150)	AC read converter block threshold value
NR-EXT	1, 2, 3, 4, 5	“Critical” extent threshold for listing file
STATINTV	1–9999 seconds (60)	Statistics-gathering interval

To change the defaults, you must edit the Natural ADVUEX1 program and make the changes directly within the program listing in the defaults area, which looks as follows:

```
     .
     .
     .
DEFINE DATA PARAMETER USING ADVPUEX1
END-DEFINE
*
* SET THE DEFAULTS
*
AOS-END-MSG = 'Y'  (Display end-of-session message)
AOS-LOGO = 'Y'     (Online System logo display—set to 'N' for no logo display)
CPEXLIST = 'N'     (Checkpoint list control—set to 'Y' for extended checkpoint list)
NR-EXT = 4         (Critical extent threshold—1, 2, 3, 4 or 5)
MAX-AC-IOS = 150   (AC read converter block threshold)
STATINTV = 60      (Statistic-gathering time. range: 1 - 9999)
*
END
```

8

INSTALLING THE RECOVERY AID (ADARAI)

To install the Adabas Recovery Aid, it is necessary to

- allocate the recovery log;
- customize the skeleton job streams for your installation (see the *Adabas Operations Manual* for more detailed information);
- update the necessary nucleus run/utility job control to include the Recovery Aid data definition statements;
- install the Adabas/ADARAI utility configuration; and
- run ADARAI PREPARE and a save operation to begin a logging generation.

Except for customizing the skeleton job stream, the specific installation steps are as follows:

Step 1. Define and format the DD/RLOGR1 file.

Use the ADAFRM RLOGFRM function to format the RLOG.

Step 2. Add DDRLOGR1 DD or DLBL statements to the nucleus job stream and to any utilities that update or save the database and thus write to the RLOG file.

Whenever these utilities are executed while ADARAI is active in the database (that is, after the PREPARE function has been executed), the DDRLOGR1 DDDLBL statements must be included.

The following utilities update the database and therefore write to the RLOG:

ADAORD (all STORE and REORDER functions)
ADALOD (all functions)
ADAINV (all functions)
ADARES REGENERATE/BACKOUT database
ADASAV RESTORE (all functions) and RESTPLOG
ADADEF NEWWORK

The following utilities save the database and therefore write to the RLOG:

ADASAV SAVE (all functions)
ADAORD RESTRUCTURE
ADAULD

The following utility functions have an impact on recovery and therefore write to the RLOG:

ADARES PLCOPY/COPY
ADASAV MERGE

Additionally, the Adabas nucleus writes to the RLOG during startup and termination. The nucleus also writes checkpoint information to the RLOG when ADADBS or Adabas Online System functions are processed, ensuring these events are known to ADARAI for recovery processing.

Step 3. Install ADARAI on the database.

Execute the ADARAI PREPARE function. ADARAI PREPARE updates the ASSO GCB to indicate that ADARAI is installed. It also creates a control record on the RLOG file with necessary ADARAI information (number of generations, RLOG size, etc.).

Step 4. Create the first ADARAI generation.

Execute ADASAV SAVE (database) to start the logging of RLOG information. See the *Adabas Utilities Manual* for more information.

Note:
Once ADARAI is active in the database, protection logging must always be used.

INSTALLING THE ERROR HANDLING AND MESSAGE BUFFERING FEATURE

Use the following procedure to install the error handling and message buffering feature of Adabas:

1. Specify ADARUN SMGT=YES. If message buffering is to be used, also specify ADARUN MSGBUF with a value greater than zero.

   When ADARUN SMGT=YES is specified to activate the error handling tool, the initialization module ADAMXI is loaded by ADARUN and is then called during session open:

   - the error handling header/environment is initialized;
   - the message buffer is initialized if ADARUN MSGBUF is specified with a value greater than zero;
   - the error handling modules are loaded into memory by ADAIOR;
   - the Adabas module table is built;
   - any provided error handling user exit is initialized;
   - the default recovery plug-in (PIN) module ADAMXY is installed;
   - the program check and abnormal termination handlers are activated;
   - the error handling flag in the header is raised indicating a successful start.
   - the ADANI2 message is generated to indicate that error handling is active in the nucleus.

2. Decide which exits are critical (the default) and issue SMGT,XNOTCRITICAL=exit-code operator commands for those that are not critical.

3. Customize ADASMXIT if necessary, particularily if PINRSP or PINAUTOR are to be activated. Reassemble the exit and ensure that it resides in the Adabas load library or is available in a load library that is available at start-up time.

4. Decide which PINs to activate.

   The following table lists the available PINs and how to activate them:

PIN Routine	To install ...
PINAUTOR	rename NOAUTOR in the Adabas load library to PINAUTOR
PINOPRSP	rename NOOPRSP in the Adabas load library to PINOPRSP
PINRSP	issue operator command SMGT,ADDPIN=PINRSP when the nucleus is active*
PINUES	issue operator command SMGT,ADDPIN=PINUES when the nucleus is active*

***** Since PINRSP and PINUES handle some of the same response codes, perform the ADDPIN function last on the module that is to acquire control. For example, PINRSP and PINUES both handle a response code 55. If PINUES is to acquire control, the ADDPIN must be done on PINUES after PINRSP.

At this point error handling is fully operational and SMGT operator commands may be issued.

10

INSTALLING AND USING THE ADABAS MIGRATION TOOL

This chapter tells you how to install and use the Adabas migration tool.

Installation

Installation Datasets

The migration tool is distributed as part of the standard Adabas OS/390 installation package.

Load Library

Migration tool members in the Adabas load library are listed in the following table:

Member	Description
MIGS01	Batch migration tool. Rename to ADALNK and make available to any batch job that needs to use the migration tool.
MIGS02	Com-plete migration tool. Rename to ADALCO and place in a COMPLIB library for any Com-plete job that needs to use the migration tool.
MIGS03	CICS command-level migration tool. Link this with the CICS command-level stubs, name it Adabas (or whatever your CICS link module is called), and place it in the DFHRPL concatenation of any CICS job that needs to use the migration tool. A sample link job is contained in MIGCICLK.
MIGS09	Batch migration tool for use with reentrant ADALNK. Rename to ADALNK and make available to any batch job that uses a reentrant ADALNK and needs to use the migration tool.

Note that there is no requirement to relink any of these modules. However, if you do, you must be sure to leave their attributes unchanged (REUS, AMODE31).

Source Library

Migration tool members in the Adabas source library are listed in the following table:

Member	Description
MIGCICIN	Sample CICS program to initialize the migration tool.
MIGLINK	Macro used to generate the migration table.
MIGT01	Sample migration table.

The source library may also contain other members such as README files or ZAPs.

Jobs Library

Migration tool members in the Adabas jobs library are listed in the following table:

Member	Description
MIGAFP32	Example ZAP to change the names of the modules loaded by Adabas Fastpath 3.2 (FASTENV, FASTNUC, FASTPRM).
MIGAVI63	Example ZAP to change the names of the modules loaded by Adabas Vista 6.3.
MIGCIASM	Sample job to assemble and link MIGCICIN.
MIGCICLK	Sample job to link MIGS03 with the CICS command-level stubs. This example creates a CICS load module called CICMIG.
MIGTASM	Sample assembly and link job for creating the migration table.
MIGZAPCI	ZAP to change defaults for the CICS migration tool (see the section **Changing Defaults** below).
MIGZAPDE	ZAP to change default database ID and buffer sizes (see the section **Changing Defaults** below).
MIGZAPRS	ZAP to change the handling of initialization contention under Com-plete and CICS (initialization must be single-threaded in those environments). By default, the migration tool delays contending users for a second at a time until initialization has completed. You can use this ZAP to give contending users a bad response code instead.
MIGZAPSV	ZAP to change the SVC number in standard batch link modules, depending on the SVC number specified in the ADARUN parameters.

Create a Migration Table

A migration table defines

- all available link modules and the versions (if any) of Adabas Fastpath, Adabas Transaction Manager, and/or Adabas Vista linked in with them (see the section **Adabas Options** for more information).
- the databases that are to be accessed using which link module. Any call to a database not defined in the table is issued using the **default** link module. You **must** nominate a default link module.

You can use the supplied sample migration table as a base. Ensure that the assembly terminates with condition code 0. Any error in parameter specification is reported in an MNOTE and generates condition code 16.

The linked migration table must be called

- MIGT01 for batch;
- MIGT02 for Com-plete;
- MIGT03 for CICS; or
- MIGT09 for batch using a reentrant link module.

The migration table must not be linked reentrant.

Define the Link Modules

Each link module named in the migration table

- must be available to the migration tool at runtime; and
- must have the correct SVC number specified in it.

Example

You access your databases through an ADALNK version 7.4.1 using SVC 249, but database 1 is still running at version 6.2.3 accessed through an ADALNK version 6.2.3 using SVC 248.

To allow batch jobs to access all databases including database 1

1. Rename your version 7.4.1 ADALNK to

   A741LNK

2. Rename your version 6.2.3 ADALNK to

   A623LNK

3. Define both link modules in the migration table, nominating A741LNK as the default and specifying that database 1 is to be accessed through A623LNK:

```
MIGLINK TYPE=LINK,LINK=A623LNK
MIGLINK TYPE=LINK,LINK=A741LNK,DEFAULT=YES
MIGLINK DBID=1,LINK=A623LNK
MIGLINK TYPE=END
END
```

Operation

The loading environment is

- STEPLIB for batch;
- COMPLIB for Com-plete; and
- DFHRPL for CICS.

To use the migration tool, you need to have available in your loading environment

- the appropriate migration tool:
  - MIGS01 renamed ADALNK for batch;
  - MIGS02 renamed ADALCO for Com-plete;
  - MIGS03 linked as Adabas (or the name of your choice) for CICS; or
  - MIGS09 renamed ADALNK for batch using a reentrant link module.

Important:
Some products such as the Adabas nucleus and Com-plete (when running in ACCESS mode) require an unmodified ADALNK. Do not put the batch migration tool on the loading environment of any Software AG product that signs on to the Adabas SVC.

- the correct migration table:
  - MIGT01 for batch;
  - MIGT02 for Com-plete;
  - MIGT03 for CICS; or
  - MIGT09 for batch using a reentrant link module.
- all link modules named in the migration table.

When the first Adabas call is issued, the migration tool loads the migration table and all link modules named in it. If any load fails, the migration tool ABENDs with a system 0C3 and the PSW pointing to one of the following error strings:

Error Message	Explanation
MIGT0x NOT AVAILABLE	The LOAD for MIGT0x failed.
NO LINK MODULES DEFINED	No link modules are defined in the migration table.
LINK MODULE NOT FOUND	The LOAD failed for one of the link modules defined in the migration table.

When initialized, the migration tool examines every Adabas call to determine the target. If the target is defined in the migration table, the call is routed to the appropriate link module. Otherwise, the call is passed to the default link module.

Under CICS, the migration tool can also be initialized using the supplied MIGCICIN program. This may in fact be necessary if you want to force initialization during PLT processing without specifying PARMTYP=ALL in ADAGSET. MIGCICIN sends a special Adabas command IM to instruct the migration tool to initialize.

Changing Link Module SVCs Dynamically

Note:
This section applies to the standard batch migration tool (MIGS01) only.

When using the batch migration tool MIGS01, the SVC number assembled into ADALNK is not overwritten at execution time by the SVC number specified in the ADARUN parameters. Instead, you must assemble the correct SVC number into each copy of ADALNK.

This means that you cannot easily switch a batch job from a production to a test SVC, for example. You would need to maintain two ADALNKs: one with the production SVC and another with the test SVC.

Depending on the SVC number specified in the ADARUN parameters, you may be able to avoid this situation by zapping translation rules into MIGS01 so that it will overwrite the SVC numbers in the link modules. The supplied job MIGZAPSV provides an example of how to do this.

An SVC translation rule is 16 bytes long. You may zap as many as 10 of these into MIGS01.

The format of each rule is:

00xx	**ADARUN SVC to which this rule applies (2-bytes where the first byte must be X'00' and the second byte X'xx' is the relevant ADARUN SVC number in hex)**
yyxx	**2-byte pairs of SVC numbers in hex (up to 7 allowed per rule where the first byte X'yy' specifies the old and the second byte X'xx' specifies the new SVC number)**
. . .	

If the link module contains the first number, it is overwritten with the second number which is usually the same as the SVC to which the rule applies; that is, X'xx'.

The final pair must have X'00' as the first number. If the second number in the final pair is not X'00', this is treated as a default value which is applied to any link module that did not match one of the preceding pairs.

If there is no rule for the ADARUN SVC, no translation is done.

Examples

Example 1:

You have 2 SVCs: 249 (Production) and 248 (Test), but all your batch ADALNKs contain SVC 249. To zap an SVC translation rule into MIGS01:

```
00F8   This rule will be in effect if ADARUN SVC=248
F9F8   Any ADALNK assembled with SVC 249 will be changed at execution time to
       use SVC 248 instead
0000   Any other ADALNK remains  unchanged
```

Example 2:

You have 4 SVCs: 249 (old Production), 248 (old Test), 239 (new Production), 238 (new Test) and you want to ensure that all ADALNKs use the new versions of the SVCs. In addition, you want to use the Test version if ADARUN SVC=248 or 238 and the Production version if ADARUN SVC=249 or 239. To zap SVC translation rules into MIGS01:

```
00F9       rule for ADARUN SVC=249
00EF       make all ADALNKs use SVC 239
00000000               pad to next rule
00000000
00000000
00F8       rule for ADARUN SVC=248
00EE       make all ADALNKs use SVC 238
00000000               pad to next rule
00000000
00000000
00EF       rule for ADARUN SVC=239
00EF       make all ADALNKs use SVC 239
00000000               pad to next rule
00000000
00000000
00EE       rule for ADARUN SVC=238
00EE       make all ADALNKs use SVC 238
00000000               pad to next rule
00000000
00000000
```

CICS Considerations

PPT Entries

You must define a PPT entry for the migration tool itself, the migration table, and all link modules named in the migration table.

The entries for the migration tool and table should have the same options as for the link modules. The migration tool and table must both be defined as resident.

LNKENAB and LNKTRUE

If you use the task-related user exit, you must have a LNKENAB and a LNKTRUE for each Adabas link module that operates in TRUE mode.

Note:
You may not define both TRUE and non-TRUE link modules in the same migration table.

In ADAGSET, specify the name of the actual link module and **not** the name of the migration tool.

Example

In order to communicate through two SVCs, 248 and 249, you might define:

ADAEN48
ADATR48
ADABAS48

—with ADAGSET parameters

ENTPT=ADABAS48,PARMTYP=ALL,SVCNO=248,TRUENM=ADATR48…

—and

ADAEN49
ADATR49
ADABAS49

—with ADAGSET parameters

ENTPT=ADABAS49,PARMTYP=ALL,SVCNO=249,TRUENM=ADATR49…

Direct Call Interface

MIGS03 supports callers using both the DCI and the EXEC CICS LINK interface to ADABAS. MIGS03, however, always uses the DCI interface to call the various link modules.

Passing Parameters

When MIGS03 is called using

- EXEC CICS LINK, it first looks for the Adabas parameter address list in COMMAREA and then in the transaction work area (TWA).

  If neither storage area contains a parameter list, MIGS03 ABENDs S0C3 with the PSW pointing to the string

  NO ADABAS PARAMETER LIST

- the DCI, R1 must contain the address of the Adabas parameter list.

Supported Versions

The migration tool only works with command-level link modules from Adabas version 6.2 and above.

Newcopy

The migration tool records the entry-point addresses of all link modules named in the migration table.

This means that if you want to activate a new copy of an Adabas link module, you **must also activate a new copy of the migration tool MIGS03 and the migration table MIGT03.**

Similarly, if you wish to activate a new copy of MIGT03, you must also activate a new copy of the migration tool MIGS03.

On balance, it is probably safer to recycle CICS.

Migration Tool Defaults

The migration tool modules contain certain default values at fixed offsets, the same as an ordinary link module.

Changing Non-CICS Module Defaults

For the non-CICS migration tools (MIGS01, MIGS02 and MIGS09), these defaults are shown in the following table:

Offset (Hex)	ADALNK Name	Default Value	Purpose
80	LNKLOGID	0	Default DBID
86	LUINFO	0	Length of user information
9C	RVINFO	0	Length of Review buffer
A0	XITBLEN	0	Length of buffer for user exit B
A2	XITALEN	0	Length of buffer for user exit A
A4	XITAFPLN	0	Length of Adabas Fastpath buffer

The migration tool uses LNKLOGID (if not 0) to select the link module for calls that have no database ID specified. If LNKLOGID is 0, these calls are given to the default link module.

The various length fields are not used by the migration tool itself. However, it is important to set them to the same values used in your current link modules because they are used by prefetch/multifetch to calculate buffer sizes and by Com-plete to suballocate its Adabas buffers. If you have different values in different link modules, use the highest value.

Changing CICS Module Defaults

The offsets are different for CICS (MIGS03), as shown in the table below. The recommendations above also apply for CICS.

Offset (Hex)	ADAGSET Name	Default Value	Purpose
9DC	LOGID	0	Default DBID
9E2	LUINFO	0	Length of user information
A38	None	0070	Total length of a UB = (UBPFXL+UBLEN+LUINFO+LRINFO +LUSAVE+7)/8*8
A42	LUSAVE	0	Length of user exit save area
A44	LRINFO	0	Length of Review buffer
A46	LXITBA	0	Length of buffer for user exit B
A48	LXITAA	0	Length of buffer for user exit A
A4A	LADAFP	0	Length of Adabas Fastpath buffer

Adabas Options

The Adabas options Adabas Fastpath, Adabas Transaction Manager, Adabas Vista, and System Coordinator have components operating in the link module and are thus affected by the migration tool.

Common Considerations

For these three options:

- each version must have its own system file; and
- when the option's link module component loads other modules (for example, FASTENV, FASTNUC and FASTPRM in the case of Adabas Fastpath), it is necessary to zap older versions to change the names of the modules they load. Similarly, the older versions of the loaded modules must be renamed.

Option-specific Considerations

Individual options impose additional restrictions.

Adabas Fastpath

Each version of Adabas Fastpath must have a separate Asynchronous Buffer Manager.

Adabas Transaction Manager

All databases participating in a transaction must be accessible through the same link module.

Adabas Vista

All partitions of a partitioned file must be accessible through the same link module.

Online Services

Adabas Fastpath, Adabas Transaction Manager, Adabas Vista, and System Coordinator each have an online services system that are written in Natural.

Certain online functions communicate with the option's link module component by issuing Adabas calls that do not contain a meaningful DBID. The link module component recognizes such calls and satisfies them.

Because the DBID is not significant, the migration tool cannot route these commands to the appropriate link module. However, it can recognize that the commands are internal to an option and it can identify the relevant option and its version.

Having identified the option and its version, the migration tool searches the migration table for a link module that has been defined as containing the matching option version and forwards the call to that link module. If no match is found, the call is passed to the default link module.

Note that a link module may contain only one version of an option and an option version may reside in only one link module.

Generate a Link Module Migration Table

A sample link module migration table is provided in the source library for use in generating a link module migration table.

1. Specify the Link Modules

You may specify up to 10 different link modules to be used in this environment.

For each link module, you must provide the following:

```
MIGLINK TYPE=LINK,LINK=link-module-name
```

One of the link modules must be designated as the default link module by providing the DEFAULT=YES parameter:

```
MIGLINK TYPE=LINK,LINK=link-module-name,DEFAULT=YES
```

Additionally, when a link module has an Adabas option linked with it, you must indicate the option version. For example:

MIGLINK TYPE=LINK,LINK=link-module-name,AFP=vrs,AVI=vrs,ATM=vrs,COR=vrs

—for a link module containing Adabas Fastpath, Adabas Vista, Adabas Transaction Manager, and System Coordinator at the version, revision, and system maintenance (SM) levels specified by “vrs”. The same version of an option may not be specified for more than one link module.

Example:

There are four (4) link modules: one per Adabas SVC 236, 237, 249, and 254.

The link module for databases on SVC 237 is the default and includes version 712 of Adabas Fastpath and Adabas Vista.

The link module for databases on SVC 254 includes old versions of Adabas Fastpath and Adabas Vista. Note that special ZAPs are required in order to run more than one version of Adabas Fastpath or Adabas Vista in a single job.

Databases on SVCs 236 and 249 use neither Fastpath nor Vista.

```
MIGLINK TYPE=LINK,LINK=ADALNK36
MIGLINK TYPE=LINK,LINK=ADALNK37,DEFAULT=YES,AFP=712,AVI=712
MIGLINK TYPE=LINK,LINK=ADALNK54,AFP=321,AVI=631
MIGLINK TYPE=LINK,LINK=ADALNK49
```

2. Specify the Databases and Other Targets

For each database that will use a link module other than the default, you must provide:

MIGLINK DBID=dbid[,LINK=link-module-name]

You must specify at least one DBID entry. If you omit the LINK parameter, the database uses the default link module. Otherwise, it should match a previously defined link module.

You may specify up to 1000 DBIDs.

Example:

Define databases and other SVC-based targets, for example Entire System Server nodes:

- Database 131 is accessed through ADALNK54, Adabas SVC 254;
- Database 122 is accessed through ADALNK49, Adabas SVC 249;
- Database 153 is accessed through ADALNK36, Adabas SVC 236;
- Calls to any other target are directed to the default link module.

```
MIGLINK DBID=131,LINK=ADALNK54
MIGLINK DBID=122,LINK=ADALNK49
MIGLINK DBID=153,LINK=ADALNK36
```

3. Generate the Migration Table

To generate a migration table with optionally up to ‘nnn’ spare DBID entries, specify

MIGLINK TYPE=END[,SPAREDB=nnn][,WTO={YES|NO}]

Specify WTO=NO if the contents of the migration table are **not** to be written to the operator console at execution time. The default is YES.

Example:

Generate the table with 10 spare entries for zapping in “emergency” targets that should not use the default link module. Do not write the migration table contents to the operator console:

```
MIGLINK TYPE=END,SPAREDB=10,WTO=NO
END
```

Migration Table DSECT

```
MLTMLT   DSECT
MLTEYE   DS    CL8
MLTALNK  DS    F                    A(TABLE OF LINK MODULES)
MLTNLNK  DS    F                    NUMBER OF LINK MODULES
MLTADBS  DS    F                    A(TABLE OF DBIDS)
MLTNDBS  DS    F                    NUMBER OF DATABASES
MLTAUSR  DS    F                    A(TABLE OF USERIDS) (FUTURE)
MLTNUSR  DS    F                    NUMBER OF USERIDS (FUTURE)
MLTRSVD  DS    XL2                  RESERVED
MLTVRL   DS    XL2                  VERSION
MLTWTO   DS    CL1                  ISSUE WTOS OR NOT
         DS    XL27                 RESERVED FOR FUTURE USE
MLTLNK   DSECT                      LINK MODULE TABLE
MLTLKNM  DS    CL8                  NAME
MLTLKAD  DS    F                    ADDRESS
MLTAFPV  DS    XL2                  VERSION OF ADABAS FASTPATH
MLTAVIV  DS    XL2                  VERSION OF ADABAS VISTA
MLTATMV  DS    XL2                  VERSION OF ADABAS TRANS. MGR.
MLTAAFV  DS    XL2                  VERSION OF ADASAF
MLTREVV  DS    XL2                  VERSION OF REVIEW
MLTCORV  DS    XL2                  VERSION OF SYSTEM COORDINATOR
MLTLKST  DS    XL1                  LINK MODULE STATUS
MLTLKOK  EQU   0                    LINK MODULE CAN BE USED
MLTLXX1  DS    XL5                  SPARE
MLTLKX   DS    XL2                  ENTRY NUMBER
MLTLLEN  EQU   *-MLTLKNM            LENGTH OF AN ENTRY
MLTDBS   DSECT                      DBID TABLE
MLTDBID  DS    XL2                  DBID
MLTDLKX  DS    XL2                  INDEX INTO LINK TABLE
MLTDLNK  DS    CL8                  LINK MODULE FOR THIS DBID
MLTDXX1  DS    XL20                 SPARE
MLTDLEN  EQU   *-MLTDBID            LENGTH OF AN ENTRY
MLTUIDS  DSECT                      USERID TABLE (FOR FUTURE USE)
MLTUSER  DS    CL8                  USERID
MLTULKX  DS    XL2                  INDEX INTO LINK TABLE
MLTULNK  DS    CL8                  LINK MODULE FOR THIS USER
MLTUXX1  DS    XL14                 SPARE
MLTULEN  EQU   *-MLTUIDS            LENGTH OF AN ENTRY
```

11

ADABAS DUMP FORMATTING TOOL (ADAFDP)

ADAFDP is the address space dump formatting module. During abnormal shutdown of the Adabas nucleus, this module receives control to format and display information that should help you analyze the reason for the error(s).

During a nucleus shutdown, ADAMPM determines the reason. If the reason is abnormal termination, ADAMPM loads the ADAFDP module into the address space prior to the 20 call to the Adabas SVC. ADAFDP subsequently receives control to format nucleus information.

If ADAFDP cannot be loaded, a message ADAF03 is written to the console and abnormal shutdown continues.

Information Formatted by ADAFDP

Much of the information formatted by ADAFDP is self-explanatory. However, because the type and amount of information depends on the shutdown situation, a brief discussion of ADAFDP output is presented.

The following is a breakdown of the information formatted by ADAFDP with a brief description of its content:

1 ADAH51 / ADAH52

The message is displayed on the console and written to DDPRINT at the point where the format begins and terminates.

2 ADAMPM ABEND CODE and PSW

If an ABEND code and program status word (PSW) were saved in ADAMPM by the Adabas ESTAE, ADAFDP displays these. In addition, ADAFDP determines the module whose entry point best fits the PSW and calculates the offset within that module. If the ADAMPM abend code and PSW are zero, ADAFDP does not format this information.

3 Adabas MODULE LOCATIONS

ADAFDP formats and displays the location of each of the Adabas nucleus modules resident in the address space.

4 ADDRESS LOCATIONS FOR USER EXITS

ADAFDP formats and displays the location of any user exit loaded with the Adabas nucleus.

5 ADDRESS LOCATIONS FOR HYPEREXITS

ADAFDP formats and displays the location of any hyperexit loaded with the Adabas nucleus. Hyperexits 10–31 are displayed as A–U, respectively.

6 ADANC0 STANDARD REGISTER SAVE AREA

Registers 0–7/8–F, which are saved in ADANC0. ADAFDP determines if any of these registers contains an address that points at a nucleus pool in storage. If yes, ADAFDP indicates which pool and snaps storage at that address. If the register is 12 and it points to a user thread, ADAFDP snaps the entire thread.

7 ADANC0 ABEND SAVE REGISTERS

Registers 0–7/8–F, which are saved in ADANC0 as a result of a user abend. ADAFDP determines if any of these saved registers contains an address that points at a nucleus pool in storage. If yes, ADAFDP indicates which pool and snaps storage at that location. If the saved register is 12 and it points to a user thread, ADAFDP snaps the entire thread.

8 ADAMPM SAVE REGISTERS

Registers 0–7/8–F, which were saved in ADAMPM by the Adabas ESTAE. These are the same registers displayed with the ADAM99 message. ADAFDP determines if any of these saved registers contains an address that points within a nucleus pool in storage. If yes, ADAFDP indicates which pool and snaps storage at that location.

9 BEGIN / ENDING ADDRESSES OF POOLS / TABLES

ADAFDP determines begin/ending address locations for pools and tables for the Adabas nucleus. These addresses are presented for easy location in the actual dump. The following is a list of pool abbreviations with explanations:

LOG	Log area
OPR	Adabas nucleus operator command processing area
CQ	Address of the command queue, which is formatted later by ADAFDP
ICQ	Internal command queue
TT	Thread table
IA1	Software AG internal area 1
SFT	Session file table
FU	File usage table
FUP	File update table
IOT	I/O table for asynchronous buffer flushing
PL2	PLOG area for asynchronous buffer flushing
PET	Table of posted ETs
TPT	Tpost
TPL	Tplatz
UQP	Unique descriptor pool
UHQ	Upper hold queue
HQ	Hold queue
UUQ	Upper user queue
UQ	User queue
FP	Format pool
FHF	File HILF element
PA	Protection area
TBI	Table of ISNs
TBQ	Table of sequential searches
WK3	Work part 3 space allocation table
IA2	Software AG internal area 2
WK2	Work part 2 space allocation table
VOL	VOLSER table
WIO	Work block I/O area
FST	Free space table work area
UT	User threads
WP	Work pool

AW2	Work block asynchronous I/O area
IOP	I/O pool related to asynchronous buffer flush
IU2	Buffer pool importance header upper 2
IU1	Buffer pool importance header upper 1
BU2	Buffer pool upper header 2
BU1	Buffer pool upper header 1
BH	Address location of the buffer pool header, information from the buffer pool header is formatted later by ADAFDP
BP	Address location of the physical start of the buffer pool

10 Adabas THREADS

ADAFDP formats the physical threads including threads 0, –1, and –2. The number of lines depends on the value of NT. The thread that was active at the time of the abnormal termination (if any) is marked by a pointer "—>".

11 USER THREADS

For any of the threads –2 to NT that had assigned work to perform, ADAFDP formats and displays information about the status of that thread. The following is a list of the formatted information:

Thread Number	–2 to NT.	
Status	Indicates the current status of the thread. The following statuses are possible:	
	Active	The currently active thread
	In Use	Thread has been assigned work
	Waiting For I/O	Waiting for a block not in buffer pool
	Waiting For RABN	Waiting for a RABN already in use
	Waiting For Work–2 Area Block	Similar to Waiting For I/O
	Waiting Workpool Space	Provides number of bytes in decimal
	Ready To Run	Waiting to be selected for execution
CMD	The Adabas command being executed.	
Response Code	Response code (if any).	

File Number	File number for this command.	
ISN	Internal sequence number for this command.	
Sub. Rsp	Subroutine response code (if any).	
Last RABN for I/O	Last RABN required by command processing, in decimal.	
Type	Last RABN type (A – ASSO, D – DATA).	
CQE Addr	Command queue element address for this command.	
User Jobname	Job name for user who executed this command.	
ITID	Internal Adabas ID for user who executed this command.	
User	User ID for user who executed this command.	
Unique global ID	28-byte ID for user who owns this command.	
Buffer Addresses	for:	control block
		format buffer
		search buffer
		value buffer
		ISN buffer
Buffer Lengths	FL	format buffer length
	RL	record buffer length
	SL	search buffer length
	VL	value buffer length
	IL	ISN buffer length
Snap Thread	The first 144 bytes of the user thread are snapped.	

12 FOLLOWING COMMANDS WERE FOUND IN THE CMD QUEUE

ADAFDP scans the command queue and formats information for any command found in the queue. If there are no commands in the command queue, ADAFDP skips. The following is a definition of the information formatted:

CQE Address	The address location of this CQE.	
F	Command queue flag bytes.	
	First Byte: General Purpose Flag	
	X’80’	User buffers in service partition, region, address space
	X’40’	ET command waiting for 12 call
	X’20’	Waiting for 16 call

	X’10’	16 call required
	X’08’	Attached buffer
	X’04’	Attached buffer required
	X’02’	X–memory lock held (MVS only)
	Second Byte: Selection Flag	
	X’80’	In process
	X’40’	Ready to be selected
	X’20’	Search for UQE done
	X’10’	UQE found
	X’08’	Not selectable during BSS=x’80’ status
	X’04’	Not selectable during ET–SYNC
	X’02’	Waiting for space
	X’01’	Waiting for ISN in HQ
CMD	The command type.	
File Number	The file number for this command.	
Job Name	Job name for the user.	
Addr User UQE	Address of users UQE, if searched for and found.	
Addr User ASCB	Address location of user’s ASCB.	
Addr ECB	Address location of user’s ECB (in user’s address space).	
Addr User UB	Address of users UB (in user’s address space).	
Addr User PAL	Address location of user’s parameter address list.	
CQE ACA	ACA field of CQE.	
CQE RQST	RQST field of CQE.	
Abuf/Pal	Address of the attached buffer/parameter address list (PAL) for CMD.	
Comm Id	28-byte unique user ID for this command.	

13 POOL INTEGRITY CHECK

ADAFDP check the integrity of several pools within the Adabas nucleus address space. If an error is detected within that pool, ADAFDP indicates which pool and what type of error was encountered. In addition, ADAFDP snaps storage at the location where the error was detected.

14 FOLLOWING RABNS / FILES ACTIVE IN BUFFER POOL

ADAFDP scans the buffer pool header for RABNs that were active or being updated. A list of those RABNs is formatted as follows:

RABN Number	The RABN number in decimal.	
Type	Type of block (A – ASSO, D – DATA).	
Flag	BP header element flag byte.	
	AKZ	X’40’ Active indicator
	UKZ	X’20’ Update indicator
	RKZ	X’10’ Read indicator
	XKZ	X’04’ Access is waiting for block
	YKZ	X’02’ Update is waiting for block
	SKZ	X’01’ Write indicator
File	File number that owns this block.	
Address	Address location of block in storage.	

15 ADAIOR REGS FOUND AT OFFSET X’080’

Registers 0–7/8–F found saved in ADAIOR at this offset. If ADAFDP determines that any of these register values is pointing within an Adabas pool, it snaps storage at that location.

16 ADAIOR REGS FOUND AT OFFSET X’0C0’

Registers 0–7/8–F found saved in ADAIOR at this offset. If ADAFDP determines that any of these register values is pointing within an Adabas pool, it snaps storage at that location.

17 ICCB POINTED FROM X’A0’ IN IOR

The ICCB address to which this offset in ADAIOR points.

18 ADAI22 ADAIOR TRACE TABLE

Format of ADAIOR’s trace table; same as that found with the ADAM99 message.

APPENDIX A : SUPPLIED TRANSLATION TABLES

Adabas EBCDIC to ASCII and ASCII to EBCDIC

```
cUES2ASC  DS    0F
c*              .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c         DC    x'000102033F093F7F3F3F3F0B0C0D0E0F'   0.
c         DC    x'101112133F3F083F18193F3F3F1D3F1F'   1.
c         DC    x'3F3F1C3F3F0A171B3F3F3F3F3F050607'   2.
c         DC    x'3F3F163F3F1E3F043F3F3F3F14153F1A'   3.
c         DC    x'203F3F3F3F3F3F3F3F3F3F2E3C282B3F'   4.
c         DC    x'263F3F3F3F3F3F3F3F3F21242A293B5E'   5.
c         DC    x'2D2F3F3F3F3F3F3F3F3F7C2C255F3E3F'   6.
c         DC    x'3F3F3F3F3F3F3F3F3F603A2340273D22'   7.
c         DC    x'3F6162636465666768693F3F3F3F3F3F'   8.
c         DC    x'3F6A6B6C6D6E6F7071723F3F3F3F3F3F'   9.
c         DC    x'3F7E737475767778797A3F3F3F5B3F3F'   A.
c         DC    x'3F3F3F3F3F3F3F3F3F3F3F3F5D3F3F'   B.
c         DC    x'7B41424344454647484 93F3F3F3F3F3F'   C.
c         DC    x'7D4A4B4C4D4E4F5051523F3F3F3F3F3F'   D.
c         DC    x'5C3F535455565758595A3F3F3F3F3F3F'   E.
c         DC    x'30313233343536373839 3F3F3F3F3F'   F.
c*              .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
          END
cUES2EBC  DS    0F
c*              .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c         DC    x'00010203372D2E2F1605250B0C0D0E0F'   0.
c         DC    x'101112133C3D322618193F27221D351F'   1.
c         DC    x'405A7F7B5B6C507D4D5D5C4E6B604B61'   2.
c         DC    x'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F'   3.
c         DC    x'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'   4.
c         DC    x'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D'   5.
c         DC    x'79818283848586878889919293949596'   6.
c         DC    x'979899A2A3A4A5A6A7A8A9C06AD0A107'   7.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   8.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   9.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   A.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   B.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   C.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   D.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   E.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   F.
c*              .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
          END
```

Entire Net-Work EBCDIC to ASCII and ASCII to EBCDIC

```
NW2ASC   DS    0F
*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
         DC    X'000102030405060708090A0B0C0D0E0F' 0.
         DC    X'101112131415161718191A1B1C1D1E1F' 1.
         DC    X'00000000000000000000000000000000' 2.
         DC    X'00000000000000000000000000000000' 3.
         DC    X'2000000000000000000005B2E3C282B5D' 4.
         DC    X'26000000000000000000021242A293B5E' 5.
         DC    X'2D2F00000000000000007C2C255F3E3F' 6.
         DC    X'00000000000000000603A2340273D22' 7.
         DC    X'006162636465666768690000000000000' 8.
         DC    X'006A6B6C6D6E6F707172000000000000' 9.
         DC    X'007E737475767778797A00005B000000' A.
         DC    X'00000000000000000000000005D0000' B.
         DC    X'7B41424344454647484900000000000' C.
         DC    X'7D4A4B4C4D4E4F505152000000000000' D.
         DC    X'5C7E535455565758595A000000000000' E.
         DC    X'303132333435363738397C00000000FF' F.
*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
NW2EBC   DS    0F
*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
         DC    X'000102030405060708090A0B0C0D0E0F' 0.
         DC    X'101112131415161718191A1B1C1D1E1F' 1.
         DC    X'405A7F7B5B6C507D4D5D5C4E6B604B61' 2.
         DC    X'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F' 3.
         DC    X'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' 4.
         DC    X'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D' 5.
         DC    X'79818283848586878889919293949596' 6.
         DC    X'979899A2A3A4A5A6A7A8A9C06AD0A100' 7.
         DC    X'00000000000000000000000000000000' 8.
         DC    X'00000000000000000000000000000000' 9.
         DC    X'00000000000000000000000000000000' A.
         DC    X'00000000000000000000000000000000' B.
         DC    X'00000000000000000000000000000000' C.
         DC    X'00000000000000000000000000000000' D.
         DC    X'00000000000000000000000000000000' E.
         DC    X'000000000000000000000000000000FF' F.
*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
         END
```

B

APPENDIX B : RELATIVE ADABAS BLOCK NUMBER(RABN) CALCULATION

Adabas identifies individual physical blocks within a given database component (Associator, Data Storage, Work) by using a relative Adabas block number (RABN).

The physical blocks within each database component are numbered in consecutive sequence beginning with 1. If the component consists of more than one physical extent (as defined by the operating system), the block numbering is continued across physical extents.

The first track of the first physical extent of the Associator, Data Storage and Work components is not used. The first track of the second and each subsequent physical extent as well as all extents of TEMP, SORT, CLOG, and PLOG is used.

The number of Adabas blocks that can be stored on a given physical unit (track/cylinder/volume) of external storage is different for each database component and for each device type (see appendix A).

Using appendix A, the number of blocks that can be stored on a given volume may be calculated as shown below:

Example 1

Associator database component, model 3380 (880 cylinders are assumed to be available on the volume).

number of ASSO blocks = blocks/track · tracks/cylinder · number of cylinders
= 19 · 15 · 880
= 250,800 Associator blocks

19 blocks must be subtracted for the first track of the first Associator physical extent; therefore, the first Associator volume can contain a maximum of 250,781 blocks.

Example 2

Data Storage database component, model 3370 (748 cylinders are assumed to be available on the volume).

number of DATA blocks = blocks/track · tracks/cylinder · number of cylinders
= 10 · 12 · 748
= 89,760 Data Storage blocks

10 blocks must be subtracted for the first track of the first Data Storage physical extent; therefore, the first Data Storage volume can contain a maximum of 89,750 blocks.

The RABN ranges stored on each VOLSER can easily be determined using the Adabas Online System report function.

C

APPENDIX C : GLOSSARY

The following terms are used in this manual to describe Adabas preparation and installation.

Adalink

The teleprocessing-monitor-dependent interface module that connects the application/user to Adabas. The actual module name depends on the environment being used; for example, the module name for linking to a batch or TSO program is ADALNK, and for CICS, the module name is ADALNC. The term “Adalink” refers to the module appropriate for the given environment.

address converter

Adabas stores each database record in a Data Storage block having a relative Adabas block number (RABN). This RABN location is kept in a table called the address converter. The address converters, one for each database file, are stored in the Associator. Address converter entries are in ISN order (that is, the first entry tells the RABN location of data for ISN 1, the 15th entry holds the RABN location of data for ISN 15, and so on).

address space

The storage area assigned to a program task/work unit. In MVS, an address space is a region; in VSE, a partition; and in BS2000, a task. In this manual, the term “region” is used as a synonym for “partition” and “task”.

communicator

A routine for communicating between operating systems, making remote targets accessible. Entire Net-work is a communicator.

database administrator

Controls and manages the database resources. Tasks include defining database distribution structure and resources, creating and maintaining programming and operation standards, ensuring high performance, resolving user problems, defining and teaching user training, controlling database access and security, and planning for growth and the integration of new database resource applications and system upgrades. Also known as the database analyst.

ID

An abbreviation of “target ID”, a unique identifier used for directing Adabas calls to their targets.

ID table

A reference data list maintained for all active targets within the boundaries of one operating system. The ID table is located in commonly addressable storage.

IIBS

The “isolated ID bit string”, a 256-bit (32-byte) string contained in the ID table header. Each bit corresponds in ascending order to a logical ID. If the bit is “1”, the corresponding ID is isolated.

isolated ID

The ID of an isolated target, which can be specified by the user as a logical ID. An isolated ID must be greater than zero and less than 256. The isolated ID is interpreted as a physical ID for addressing the target.

isolated target

A target called directly by a user.

logical ID

A user’s identifier of target(s) to which a message is directed. It must be greater than zero (0) and less than 256 (either explicitly or implicitly, the content of the first byte of ACBFNR is a logical ID).

non-DB target

A target that is not an Adabas nucleus. Access and X-COM are non-DB targets.

physical ID

The identifier of a target. It must be greater than zero (0) and less than 65,536. A database ID (DBID) is a physical ID.

pseudo-cylinder

The logical cylinder on an fixed-block-addressed (FBA) device that has no actual DASD cylinder.

reset

A flag bit is said to be reset when it contains 0.

router

A central routine for communication within the boundaries of one operating system. The routine is called by users with Adalink routines, and by targets with ADAMPM. The router's main purpose is to transfer information between the Adalink and Adabas. The router also maintains the ID table. VM/ESA, z/VM, and BS2000 environments divide router functions among Adalink or other Adabas functions. The Adabas SVCs in OS/390, z/OS, and VSE/ESA are examples of routers.

service

A processor of Adabas calls and issuer of replies. An Adabas nucleus is an example of a service (see also target).

set

A flag bit is said to be set when it contains 1.

target

A receiver of Adabas calls. A target maintains a command queue, and communicates with routers using ADAMPM. A target is also classified as a service (see definition). The Adabas nucleus is a target.

user

A batch or online application program that generates Adabas calls and uses an Adalink for communication.

INDEX

A

B

C

D

E

F

I

L

M

N

O

P

Q

R

S

T

U

V

X

Z